I Speak BASIC to *My* TI-99/4A™

Aubrey B. Jones, Jr.

A field-tested computer literacy course that introduces students to BASIC language programming.

HAYDEN

I Speak BASIC to My TI-99/4A™

I Speak BASIC to My TI-99/4A™

Aubrey B. Jones, Jr.

HAYDEN BOOK COMPANY, INC.
Hasbrouck Heights, New Jersey

To Alyce, Aubrey III, and Adrienne

Production Editor: LORI WILLIAMS
Developmental Editor: KAREN PASTUZYN
Book Design: JOHN M-RÖBLIN
Compositor: VAN GROUW COMPOSITION CO., INC.
Printed and bound by: COMMAND WEB OFFSET INC.

Library of Congress Cataloging in Publication Data

Jones, Aubrey B.
I speak BASIC to my TI-99/4A.

1. TI-99/4A (Computer)—Programming. 2. BASIC (Computer program language) I. Title. II. Title: I speak BASIC to my TI-99/4A.
QA76.8.T133J66 1984b 001.64'2 83-26674
ISBN 0-8104-6173-0

1	2	3	4	5	6	7	8	9	PRINTING
84	85	86	87	88	89	90	91	92	YEAR

Preface

Welcome to the world of computers. If you are a person who has recently bought a computer or who has use of a computer, this book can help you. It was developed for those who have little or no understanding of computers but who would like to learn more about them.

The computer is a valuable tool that can solve problems, print words, draw pictures, store information, play games, and do many other things. But it is important to understand that a computer without the proper programs (instructions) is of very little value to you. This book teaches you how to write programs in BASIC, a very simple, easy-to-use (and learn) programming language.

Computers are found almost everywhere today — video games, appliances (microwave oven control), autos (mileage computers), and cameras (automatic exposure). With the cost of computers continuing to fall, especially the very small microcomputers, they will soon become as ubiquitous as television sets. You see them at work, at the bank, and at home. In the future, they will be as commonplace and as simple to use as telephones.

Because of the influence computers will have on our society, it is important that you prepare yourself to meet the challenges of the future. I recognize that you might not want to become a programmer, an engineer, or a systems analyst, but I feel that you should become computer literate. To become computer literate, by my definition, means that you understand the limitations of a computer. It means that you are not intimidated if you are asked to use a computer. It means that you recognize that the computer is a tool, a very powerful tool, that can help you become more productive and effective in our society.

This book will be just the beginning of a very exciting and rewarding experience with computers for you. Even if you learn just a few of the fundamentals about computers and learn how to write only very simple programs, you should remember this: *You do not need to know how to program a computer in order to use one!* That is, you could purchase programs written by someone else from most computer stores. But I feel that you will have a better appreciation for the computer if you understand a little about how programs are written and how the computer (together with its programs) provides you with the desired results.

Finally, this book will get you started with computers, will eliminate the mystique of computers, will help you build your self-confidence if you need it, and (best of all) will be a lot of fun. After you complete this book, you will be able to say, "I speak BASIC."

Aubrey B. Jones, Jr.

CONTENTS

Part 1 **Hardware (The Machines)** **1**
Objectives; Typical Data Processing Operation: Basic Parts of a Computer; Box Diagrams of Computer and Microcomputer Systems; Practice 1

Part 2 **Software (The "Program")** **13**
Objectives; How Humans Talk to Computers; Machine Language (Bit, Byte, KByte); A Program in BASIC; TI-99/4A Keyboard; TI-99/4A Power-Up Rules; Practice 2

Part 3 **Your First Computer Program** **35**
Objectives; Writing Your First Computer Program; Executing Your Program; Correcting Common Errors; Key Words (PRINT, REM, END); Commands (CALL CLEAR, LIST, NEW, RUN); Edit Mode; Practices 3, 4, 5

Part 4 **More Programming Tools** **61**
Objectives; Math Operators; Order of Arithmetic Operations; Variables; Key Word (LET); PRINT Zones; Use of Commas and Semicolons in PRINT Statements; Practices 6, 7

Part 5 **Scientific Notation** **83**
Objectives; Scientific Notation; Scientific Notation on the TI-99/4A; Review and Feedback; Practice 8

Part 6 **Relational Operators and IF-THEN/GOTO Statements** **91**
Objectives; Definition of Relational Operators; Key Words (IF-THEN); Using IF-THEN Statements (Conditional Branching); A Counting Program; Key Word (GOTO); Using GOTO Statements (Unconditional Branching); Practices 9, 10

Part 7 **The INPUT Statement** **103**
Objectives; Key Word (INPUT); Using INPUT Statements; String Variables; Practices 11, 12, 13

Part 8 **Using the Calculator Mode** **117**
Objectives; Order of Operations (Review); Using the Calculator or Immediate Mode; Command (NEW); Practice 14

Part 9 **Using the Cassette Recorder** **123**
Objectives; Commands (SAVE, OLD); Using the Cassette Recorder as an Input/Output Device; Check Tape Option; Practices 15, 16, 17

Part 10 **Using FOR-NEXT...STEP Statements** **131**
Objectives; Key Words (FOR-NEXT...STEP); Comparison of GOTO, IF-THEN, and FOR-NEXT Program Loops; Loop Flowcharts; Timer Loops; Practices 18, 19

Part 11 **Reading Data** **147**
Objectives; Key Words (READ, DATA); Using READ-DATA Statements; Key Word (RESTORE); Practice 20

Part 12 **Video Display Graphics** **159**
Objectives; Video Display Layout; Video Display Worksheets; Key Words (CALL HCHAR, CALL VCHAR, CALL CHAR); Character Codes; Using Graphic Commands; Formatting Output Using TAB Function; Practice 21

Part 13 **Arrays** **179**
Objectives; Definition of an Array; Program Examples Using One- and Two-Dimensional Arrays; Key Word (DIM); Checkbook Array Program; Practices 22, 23

Part 14 **INT(X), ABS(X), and RND Functions** **193**
Objectives; INT(X) Function; ABS(X) Function; RND Function; RANDOMIZE Statement; Coin Toss Program; Guess the Number Program; Practices 24, 25

Part 15 **Subroutines** **207**
Objectives; Definition of a Subroutine; Key Words (GOSUB, ON-GOTO, ON-GOSUB); Using Subroutines; Temperature Conversion Program; A Quiz Program; Practices 26, 27

Extra Practices **224**

I Speak BASIC to My TI-99/4A™

PART 1

Hardware (The Machines)

What You Will Learn

1. That the computer is a valuable tool that can solve problems, print words, draw pictures, store information, retrieve information, compare information, play games, and do many other things to help you in everyday life.
2. That people control computers and that computers cannot think (despite what you might have heard).
3. To identify and explain the basic parts of a computer and relate them to a "box diagram" of a general purpose computer.
4. To identify and explain the function of the basic parts of a TI-99/4A microcomputer.
5. To define and explain the terms hardware, software, microcomputer, microprocessor, RAM, ROM, processor, input unit, output unit, and memory.
6. That computers are simple and easy to use; and above all that computers are fun!

Welcome to the World of Computers

People Control Computers!

Computers Can't Think!

- Let's destroy some myths. First of all, despite what you might have heard, people control computers, people design them, people build them, people sell them, and, most of all, people tell them what to do (which is another way of saying that people "program" them).
- A computer program is a set of instructions that specify what the computer must do. Computer programmers write these instructions.

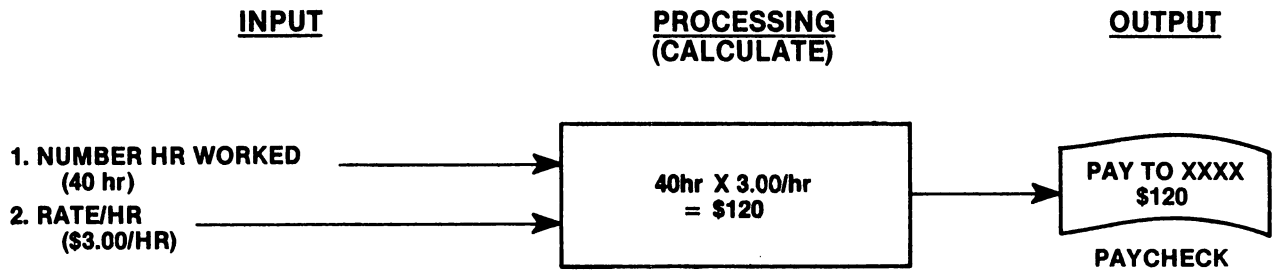
1.1

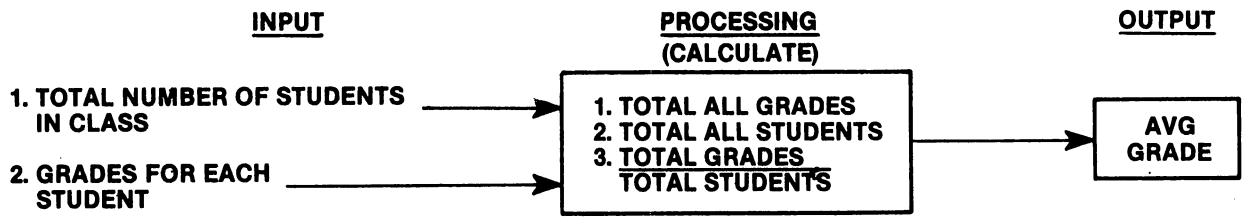
Box Diagram of Typical Data Processing Operation

1.2

Examples of Data Processing Operation

- Data are collections or representations of facts or instructions.
- Although a computer isn't necessary to perform a data processing operation, a computer system is capable of processing data at tremendous speeds.





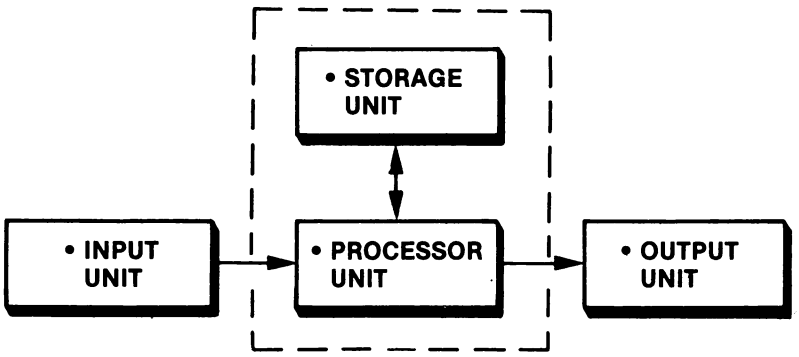
1.3

Terms You Should Know

- **HARDWARE**
  — The computer and computer-related equipment (the machines)
- **SOFTWARE**
  — The instructions for the computer (the program)

1.4

Box Diagram Showing Basic Parts of a Computer



1.5

Stores or Remembers

- **Storage unit (memory)**
  - **— Stores both information and instructions until needed (requested)**

- Computers are controlled by the program which is in the main storage unit.

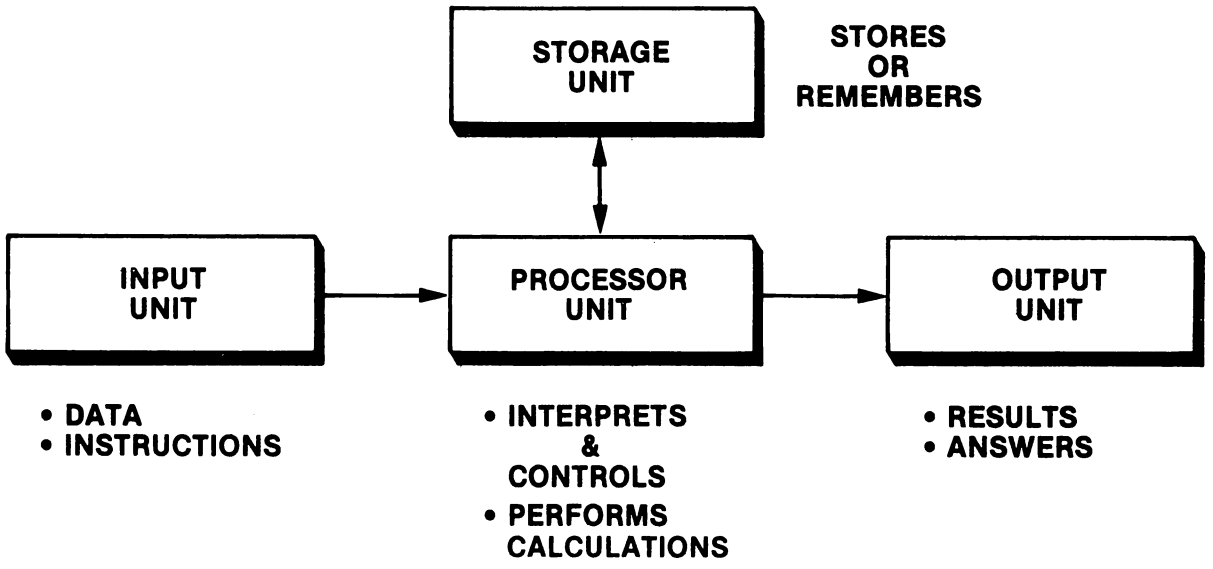
1.6

Interprets, Controls, and Calculates

- **PROCESSOR UNIT**
  - **— INTERPRETS (DECODES) INSTRUCTIONS AND REGULATES (CONTROLS) THEIR EXECUTION**
  - **— PERFORMS ALL OF THE CALCULATIONS**

1.7

Box Diagram of a Computer System



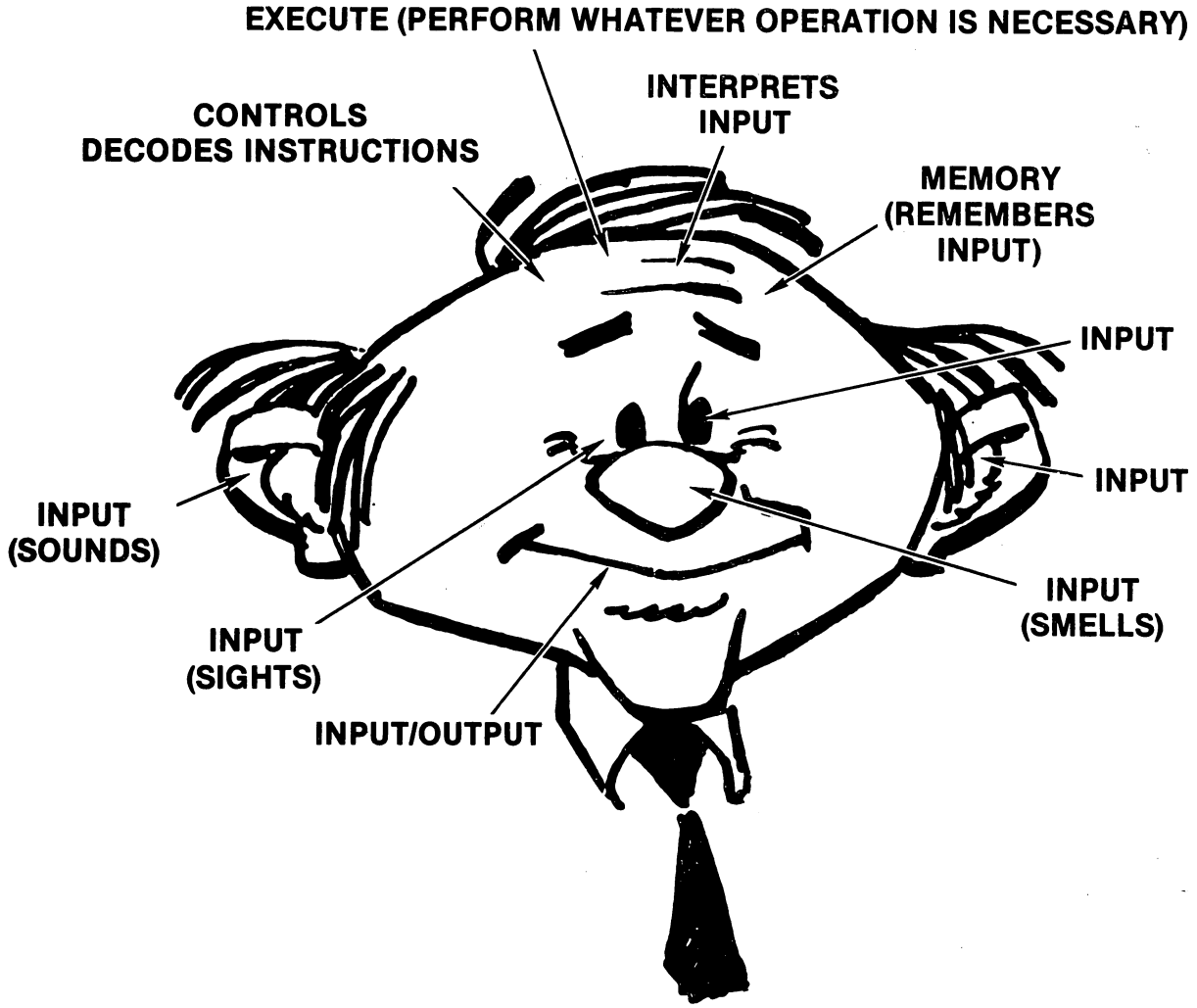
1.8

What We Have Learned

- Input → Provides instructions and data
- Storage → Stores or remembers (memory)
- Processor → Interprets, controls, and calculates
- Output → Provides answers and results

1.9

"Human Computer" Man Can Think but Computer Can't!



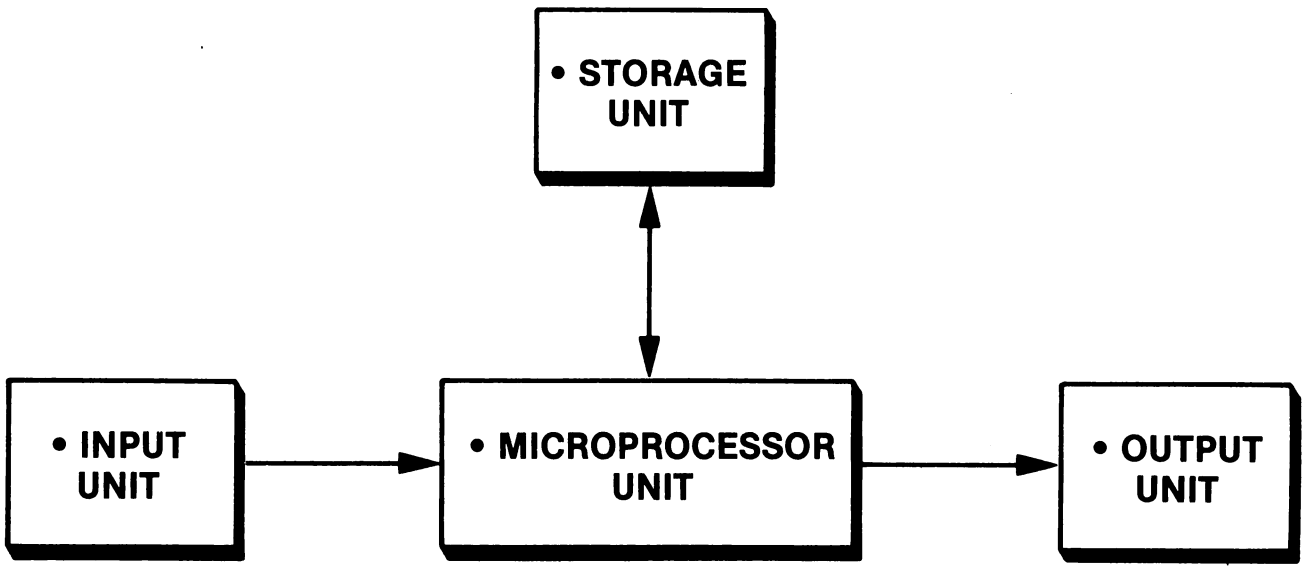
1.10

More Terms You Should Know

- **Microprocessor** = Very small processor.
- **Microcomputer** = Very small computer
- **RAM** = Random Access Memory
  — CAN be changed by the user
  — Information stored in RAM will be destroyed if power fails or is turned off (volatile)
- **ROM** = Read Only Memory
  — CANNOT be changed by the user
  — Information stored in ROM is not destroyed if power fails or is turned off (nonvolatile)
  — Control program (BASIC interpreter) stored here

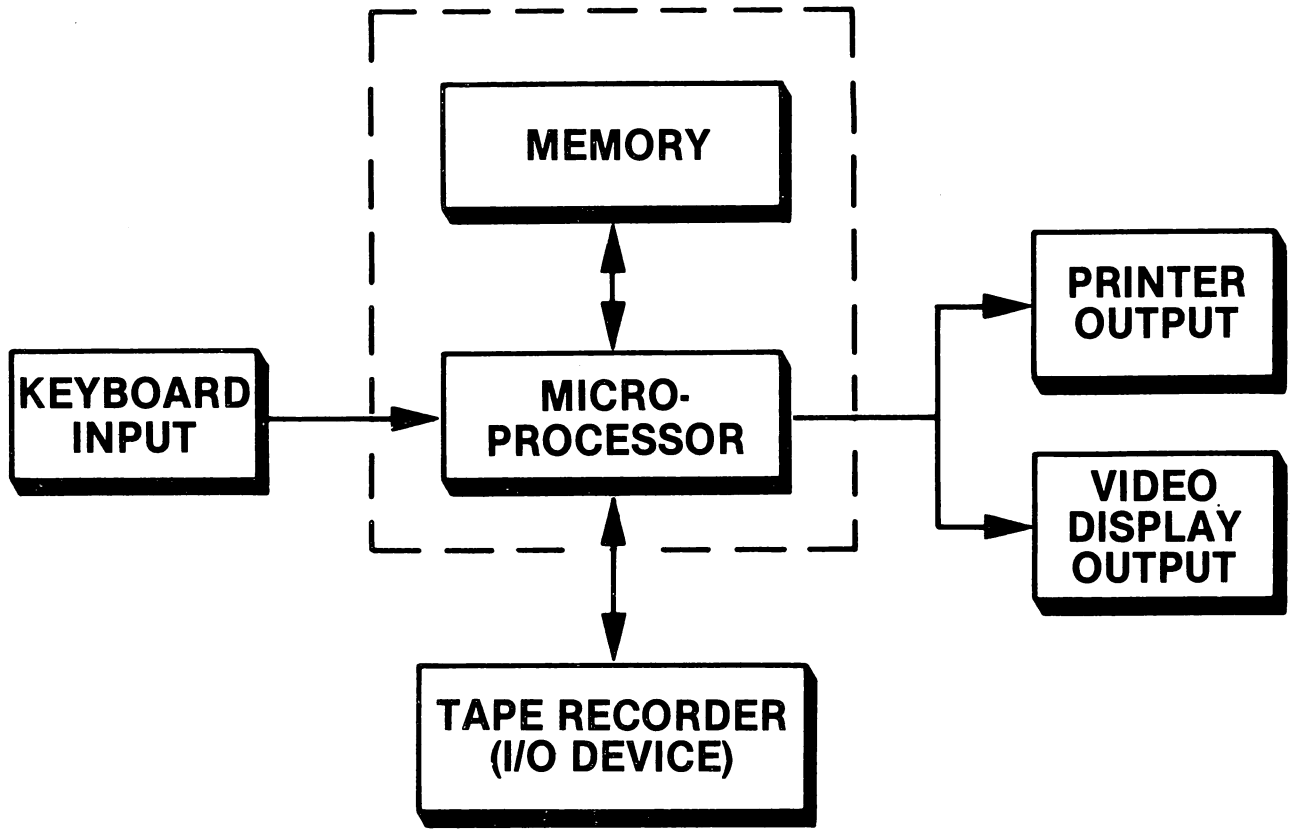
1.11

Box Diagram of a Microcomputer



1.12

Basic Components of the TI-99/4A Computer



1.13

TI-99/4A Computer System

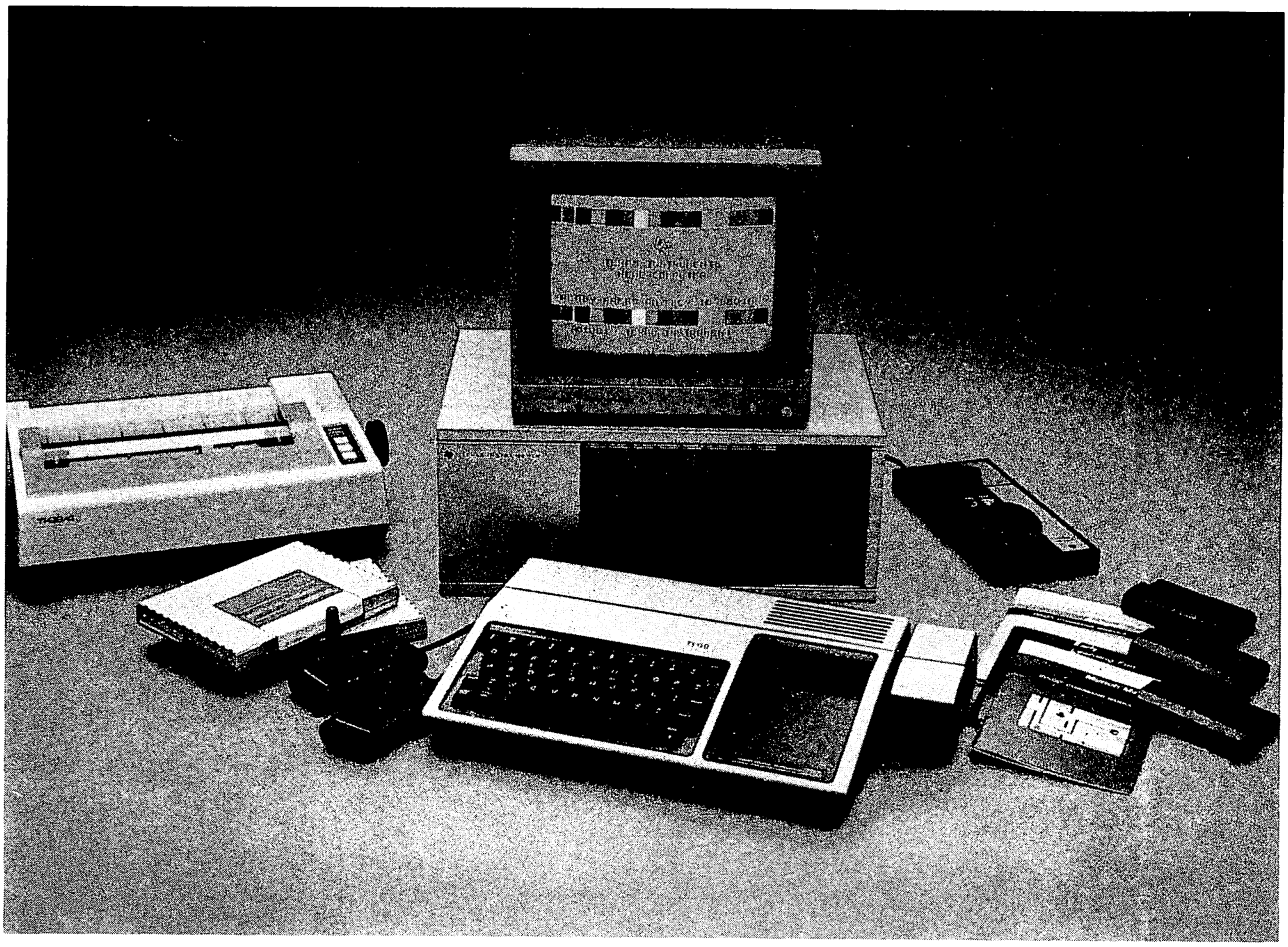


Photo Courtesy of Texas Instruments Incorporated

1.14

What We Have Learned

DATA PROCESSING OPERATION STEPS:	BASIC COMPUTER PARTS:	MICROCOMPUTER PARTS:
• INPUT →	• INPUT UNIT →	• INPUT UNIT
• PROCESSING →	• PROCESSOR UNIT + MEMORY UNIT →	• MICROPROCESSOR + MEMORY
• OUTPUT →	• OUTPUT UNIT →	• OUTPUT UNIT

PRACTICE 1

Box Diagram of a Computer

1. Draw the box diagram of a computer system.
   a. Label each box with the correct name.
   b. List the functions of each box.

PART 2

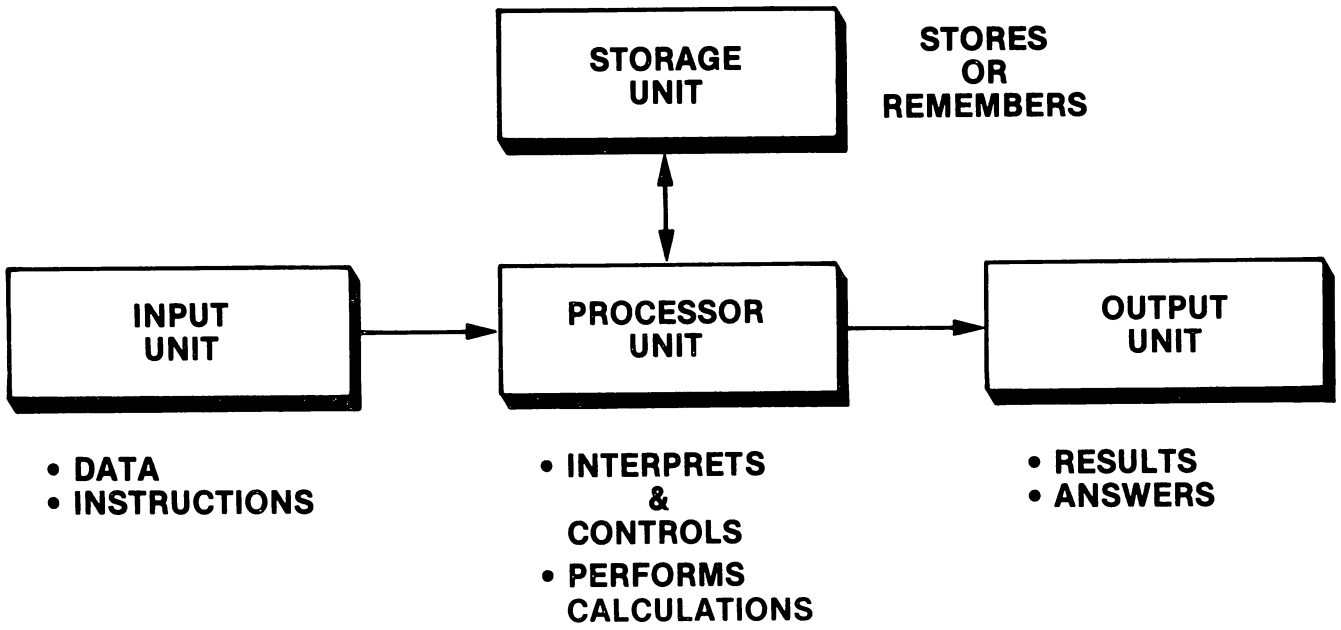
Software (The "Program")

What You Will Learn

1. To define the terms hardware, software, BASIC, binary, and interpreter, and to relate them to computers.
2. That computers speak a foreign language: machine language.
3. How humans talk to computers via a programming language called BASIC.
4. To identify the principal parts of a BASIC program.
5. To identify and explain the purpose of all the keys on the TI-99/4A keyboard.
6. How to connect and power up a TI-99/4A microcomputer.

REVIEW

Box Diagram of a Computer System



REVIEW

Terms You Should Know

- **HARDWARE**
  — The computer and computer-related equipment (the machines)
- **SOFTWARE**
  — The instructions for the computer (the program)

2.1

Computers Speak a Foreign Language! (No Speak English, French, German, Spanish, or Any Other Natural Language)

- Computers speak in **machine** language
  - — Machine language is a form of **binary** coding
  - — Binary is a word denoting "two"
  - — Machine language uses two symbols: "Ø" and "1"
- A computer is capable of executing only machine language instructions.

2.2

Machine Language: Bit

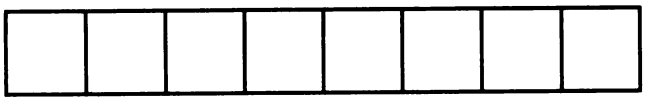
Bit = binary digit
Bit = smallest memory cell in a computer
Bit = "1" or "Ø"

- Machine language instructions are stored in the main computer storage of the processor unit.
- These instructions are coded in bits.

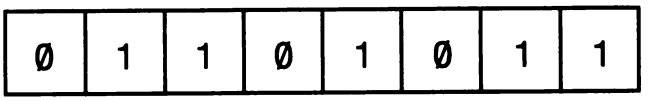
2.3

Machine Language: Byte

Memory Cell with 1 Bit



8 Memory Cells



8 Bits = 1 Byte

- A bit was found to be insufficient to store all the letters of the alphabet, special characters, and numbers needed to process data.
- A byte is a series of 8 bits.
- A byte is used to store a single letter, number, or character. For example, a byte might contain the binary equivalent of the letter "A" or the number "7."

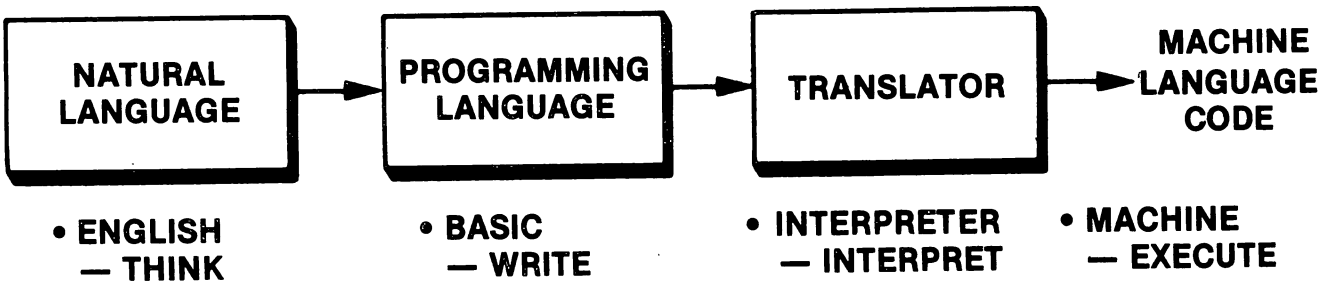
2.4

Machine Language: KByte

Byte = 8 Bits
K = 1000
KBytes = 1000 Bytes
KBytes = 8000 Bits

- More exactly, a byte = 2^{10} (1024) bits, but you need know only that K stands for "one thousand."
- Microcomputers use RAM as their main computer storage. A 32K microcomputer is one that has 32000 (or more exactly, 32768) bytes of RAM.

2.5
How Humans Talk to Computers



- Even though a computer can execute only machine language instructions, it is not necessary to learn machine language to communicate with a computer.
- People normally start to speak with their natural language, but we need a *programming language* to talk to a computer.
- A program, located in ROM in a microcomputer and called an *interpreter program*, interprets or translates a programming language into machine language.
- There are many programming languages, some of which are COBOL (**Co**mmon **B**usiness **O**riented **L**anguage), FORTRAN (**For**mula **Tran**slation), RPG (**R**eport **P**rogram **G**enerator), PL/1, Pascal, and BASIC (**B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode).

2.6

Terms You Should Know

- **BASIC**
  (Beginner's All-purpose Symbolic Instruction Code)
  — Popular programming language for writing instructions to the computer

- **Interpreter**
  — Translates BASIC into machine code
  — (You really don't have to know anything about an interpreter since it is used automatically when you run a BASIC program)
  — Located in the ROM in the TI-99/4A

2.7

A Comparison between English and BASIC

To Program You Must Learn the Language First!

English Language

- Words
  — Used to make sentences
- Sentences
  — Used to make paragraphs
- Paragraphs
  — Lengths vary
- Commands
  — Can be one word — e.g., STOP! HALT!
- Sentence Numbers
  — Optional (seldom used)

BASIC Programming Language

- Key Words
  — Used to make statements
- Statements
  — Used to make programs
- Programs
  — Lengths vary
- Commands
  — Executed immediately — e.g., NEW, LIST, RUN
- Line Numbers
  — Must be used for each statement

2.8

Learning a New Vocabulary

Here Are the Key Words and Commands You'll Learn:

Key Words

- PRINT
- END
- LET
- INPUT
- GOTO
- IF-THEN
- REM
- STOP
- FOR-NEXT
- READ-DATA

Commands

- NEW
- LIST
- RUN
- CONTINUE

2.9

Commands versus Statements

COMMANDS

— Executed as soon as you type them and press ENTER

STATEMENTS

— Put into programs and executed only after you type the command RUN and press ENTER

2.10
A Program in BASIC

	Line Number	Key Word	Other Part of the Statement	"Look at" Request Ⓐ
1st Statement	1Ø	PRINT	"HELLO THERE"	ENTER
2nd Statement	2Ø	PRINT	"YOUR NAME"	ENTER
3rd Statement	3Ø	END		ENTER
Command	RUN			ENTER

Note:

Ⓐ Pressing the ENTER key tells the computer to "look at" (and store) what you have just typed. You must press this key after each statement or command.

2.11

Line Numbers

- Serve as a guide to the computer in running the program
- Tell the computer in what order it should carry out your instructions
- Normally are multiples of 5's, 1Ø's, or some other multiples to leave space for inserting new program lines between old ones

Note:

○ Computer will start executing at lowest numbered line unless told to start elsewhere.

○ Although it is perfectly legal to number program lines more closely (like 1, 2, 3, 4, etc.), don't do it!

2.12

Key Words

- Never used alone
- Need line number
- Always part of a BASIC statement that has some other part to it
- Executed only after command RUN is typed and ENTER key is pressed

2.13
What We Have Learned

- Key words
  — Used to make statements
- Statements
  — Must have line numbers and key words
  — Used to make programs
- Programs
  — May vary in length
- Commands
  — Executed as soon as you type them and press ENTER

2.14

TI-99/4A Keyboard

Photo Courtesy of Texas Instruments Incorporated

2.15

Special Function Keys on the TI-99/4A Keyboard

KEY	FUNCTION
ENTER	• Causes the computer to "look at" the line you just typed in and to act accordingly. The key must be pressed each time you want to enter a line from the keyboard. — You may use up to four screen lines for each program line before you press [ENTER]. — [⇧] or [⇩] keys work exactly the same as the [ENTER] key except in Edit mode. (Refer to [FCTN] [E] and [FCTN] [X] below.)
ALPHA LOCK	• Pressing this key until it "clicks" into place locks all of the alphabetical keys into the Upper-case mode. The number and punctuation keys are not affected, however. To release the [ALPHA/LOCK] key, just press the key again and the keyboard returns to the Lower-case mode. — You will use the computer with the [ALPHA/LOCK] key down (Upper-case mode) for all the lessons in this book.
SHIFT	• Some keys have two characters or symbols printed on the top. Use the [SHIFT] key to type upper-case characters like the exclamation mark (!), number symbol (#), dollar sign ($), ampersand (&), asterisk (*), and percent symbol (%). The [SHIFT] key must be held down while pressing a key to obtain the shifted character of that key. — In the Lower-case mode, pressing the [SHIFT] key while pressing any letter key will display the upper-case (capital) character of that letter.
SPACE BAR	• Operates just like the space bar on a regular typewriter. When you press the [SPACE BAR], the computer leaves a blank space between words, letters, or numbers. — The [SPACE BAR] can be used to correct errors. This will be explained in the section on Editing.

KEY	FUNCTION
CTRL	• Stands for "CONTROL." This key is used primarily for telecommunications to permit entry of special control characters. To enter a control character, hold down the CTRL key and press the appropriate letter or number key. (For a list of Standard ASCII control characters, refer to the Appendix of the *User's Reference Guide.*)
FCTN	• Stands for "FUNCTION." Several keys have an additional function that is obtained by holding down the FCTN key while the other keys are pressed. — Most of the functions — "AID," "CLEAR," "QUIT," etc. — are identified on the strip overlay packed with the computer. (If you have not inserted this strip into the slot above the top row of keys on the keyboard, do it now!) — Note that certain symbols (⇧, ~, [,], ?, ", etc.) are printed on the front of several letter keys.
FCTN **1** **(DELete)**	• Holding down the FCTN key while pressing the 1 key will permit you to delete characters from the program line. To delete characters, position the cursor (using the ⇦ or ⇨ key) over the character you wish to delete, then press the FCTN and 1 keys.
FCTN **2** **(INSert)**	• Puts the computer in Insert mode and permits you to insert characters into the middle of a program line. To insert characters, position the cursor (using the ⇦ or ⇨ key) over the character immediately to the right of the place you wish to insert characters, then press the FCTN and 2 keys (the computer is now in Insert mode). After you have pressed these keys, each time you type a character, the cursor and every character of the program line that is not to the left of the cursor are moved to the right. — Each character you type is inserted into the blank position left by the shifting of the cursor and other characters. — Note that characters shifted off the end of the program line are deleted from the line. — To get out of Insert mode, press the ENTER key.
FCTN **3** **(ERASE)**	• Erases the entire program line, including the line number you are retyping, *if you have not pressed* the ENTER key. — If the computer is in Edit mode, however, the entire line displayed for editing is erased *except the line number.*

KEY	FUNCTION
FCTN 4 (CLEAR or BREAK)	• When you press the FCTN 4 keys, you can perform several functions, depending on when you press these keys. — If a program is running, pressing FCTN 4 stops the execution of the program. The message BREAKPOINT AT LINE NUMBER is displayed. To continue execution, type CONTINUE and then press ENTER. — If you are typing in a program line but do not wish to enter it, pressing the FCTN 4 keys causes the program line to scroll (move) up on the screen, but that line is not entered (because you have not pressed the ENTER key yet). — If the computer is in Edit mode, the current line scrolls up on the screen and the computer leaves Edit mode. Any changes you made on the line before pressing the FCTN 4 keys are ignored. That is, the existing program line does not change.
FCTN 5 (BEGIN) FCTN 6 (PROC'D) FCTN 7 (AID) FCTN 8 (REDO) FCTN 9 (BACK)	• These keys (BEGIN, PROC'D, AID, REDO, and BACK) have special functions with software applications and will not be covered in this book. (Applications packages or software packages are programs that have been written for the user. These packages include games, mathematics courseware, personal record keeping, and the like.
FCTN D (⇨) or (Forwardspace)	• ⇨ or cursor right key moves the cursor to the right one character position. The cursor does not erase or change the characters on the screen as it passes over them. — When the cursor reaches the end of a line on the screen, it wraps around the screen and moves to the beginning of the next line down. (Note! When the cursor reaches the end of its "logical line," which is four screen lines, pressing the ⇨ has no effect.)

KEY	FUNCTION
FCTN S (⇦) or (Backspace)	• ⇦ or cursor left moves the cursor to the left one character position. Using the ⇦ does not delete or change the character it passes over. — When the cursor reaches the end of a line, it wraps around and moves up one row and to the extreme right-hand end of the row. (Note! If the cursor reaches the beginning of the line, pressing the ⇦ key has no effect.)
FCTN E (⇧)	• The up-arrow (⇧) and down-arrow (⇩) keys work exactly like the ENTER key, except in Edit mode. (The Edit mode will be explained on subsequent pages.)
FCTN X (⇩)	
FCTN = (QUIT)	• You press these keys any time you want the computer to return to the master screen, i.e., the screen you see when the computer is first turned on. — When you press these keys, all data or program information you entered will be erased.

2.16

Symbols Displayed with SHIFT Key Held Down

These symbols appear as the upper symbols on keys with two characters:

SYMBOLS	NAME	HOLD DOWN	THEN PRESS
!	Exclamation point	SHIFT	1
@	At sign	SHIFT	2
#	Number or pound sign	SHIFT	3
$	Dollar sign	SHIFT	4
%	Percent	SHIFT	5
^	Caret (exponentiation)	SHIFT	6
&	Ampersand	SHIFT	7
*	Asterisk	SHIFT	8
(	Open parenthesis	SHIFT	9
)	Close parenthesis	SHIFT	Ø
+	Plus sign	SHIFT	=
–	Minus sign	SHIFT	/
<	Less than	SHIFT	, (comma)
>	Greater than	SHIFT	. (period)
:	Colon	SHIFT	; (semicolon)

2.17

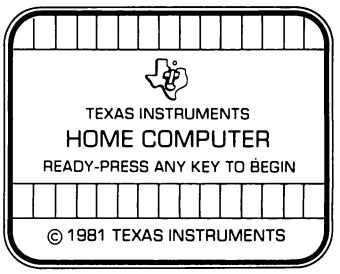
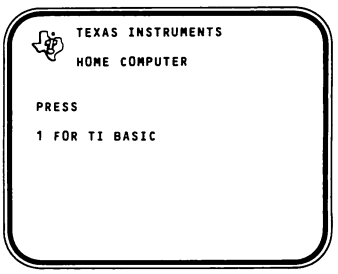
Symbols Displayed with FCTN Key Held Down

These symbols appear on the front of keys:

SYMBOLS	NAME	HOLD DOWN	THEN PRESS
~	Tilde	FCTN	W
[	Open bracket	FCTN	R
]	Close bracket	FCTN	T
___	Line	FCTN	U
?	Question mark	FCTN	I
'	Apostrophe	FCTN	O
"	Quote	FCTN	P
¦	—	FCTN	A
{	Left brace	FCTN	F
}	Right brace	FCTN	G
\	Reverse slant	FCTN	Z
\	Reverse slant	FCTN	C

2.18

TI-99/4A Power-Up Rules

YOUR ACTION	DISPLAY
1. Make certain the TI-99/4A micro-computer is connected properly (refer to the *User's Reference Guide* if you have questions).	
2. If the tape recorder is connected, it should be in the *STOP* mode. (This procedure assumes that you are not using a disk.)	
3. Turn on the video display and set the RF modulator to "Modulator." (Make certain that channel selection on the television knob matches that on the computer.)	
4. Turn on the TI-99/4A. The power switch is located on the lower-right front of the console.	
5. After a few seconds the message should appear on the screen as shown.	
6. Press any key on the keyboard and you will see:	
7. Now press [1] on the keyboard to select TI BASIC. The screen will appear as shown and you are ready to use TI BASIC.	TI BASIC READY >■ ←(cursor)

Note:

○ Whenever the flashing cursor (■) appears on the screen, the computer is waiting for you to enter something from the keyboard. The prompt character (>) marks the beginning of each line.

PRACTICE 2

Becoming Familiar with Your TI-99/4A

Become familiar with the TI-99/4A microcomputer by doing the following:

1. Turn on the TI-99/4A using the Power-Up Rules (see page 32).
2. Where is the On-Off switch for the console located? _____
3. Locate the [SHIFT] key.
   a. How many [SHIFT] keys are there on the keyboard? _____
   b. Hold down the [SHIFT] key and press every key that has a second symbol on it (e.g., pressing [1] and [2]). What happened? _____
   (Note! If you see some symbols appear on the screen, don't worry about what they are used for because you will learn about them later.)
   c. What happens if you hold down the [SHIFT] key and press [1], [3], [6], [8], and [=]? _____
4. Locate the [FCTN] key.
   a. Hold down the [FCTN] key and press every key with a symbol on the front of it except the [E] and [X] keys. What happened? _____
   b. What keys do you press to get a quote ("), question mark (?), apostrophe ('), and open bracket ([)? _____
   c. Type CALL CLEAR to clear the screen. Then move the cursor right ([⇨]) by holding down the [FCTN] key while pressing [D]. Starting at the beginning of a line (>), move the cursor until it stops. How many lines did it move before it stopped? _____
   d. Now move the cursor left ([⇦]) by holding down the [FCTN] key while pressing [S]. How far did the cursor move before it stopped? _____
   e. Hold down the [FCTN] key while pressing the [=] key. What happened? _____
5. Note that all the keys have a repeat feature that keeps the cursor moving or causes the character to repeat until you release the key. (You try it!)
6. With the computer, you cannot type the letter "L" as the number "1." Also, you should never substitute the letter "O" for the number zero. A slash (Ø) is used to help you recognize a zero on the keyboard only. The computer screen displays the letter "O" with squared "□" corners and displays a zero with rounded "O" corners to help you distinguish them. But it is still difficult to distinguish them on the screen unless you know the difference. Experiment by typing a zero and then the letter "O" until you recognize the difference on the screen.
7. Beware of the [QUIT] key! Holding down the [FCTN] key while pressing the [=] key causes the computer to erase all data or program material you have entered.

PART 3

Your First Computer Program

What You Will Learn

1. To enter and RUN your first BASIC program.
2. To explain the purpose and use of the following BASIC commands: **CALL CLEAR, LIST, NEW, RUN.**
3. To explain the purpose and use of the following key words: **PRINT, REM, END.**
4. To explain the purpose and use of the following special function keys: [CLEAR], [ENTER], [SHIFT], [⇦], [ALPHA/LOCK], [CTRL], [FCTN].
5. To explain the purpose and use of the following miscellaneous points: > (prompt), ■ (cursor), "" (quotes), line numbers, power-up rules.

REVIEW

Special Function Keys on the TI-99/4A Keyboard

KEY	FUNCTION
ENTER	• Causes the computer to "look at" the line you just typed in and to act accordingly. The key must be pressed each time you want to enter a line from the keyboard. — You may use up to four screen lines for each program line before you press [ENTER]. — [⇧] or [⇩] keys work exactly the same as the [ENTER] key except in Edit mode. (Refer to [FCTN] [E] and [FCTN] [X] below.)
ALPHA LOCK	• Pressing this key until it "clicks" into place locks all of the alphabetical keys into the Upper-case mode. The number and punctuation keys are not affected, however. To release the [ALPHA/LOCK] key, just press the key again and the keyboard returns to the Lower-case mode. — You will use the computer with the [ALPHA/LOCK] key down (Upper-case mode) for all the lessons in this book.
SHIFT	• Some keys have two characters or symbols printed on the top. Use the [SHIFT] key to type upper-case characters like the exclamation mark (!), number symbol (#), dollar sign ($), ampersand (&), asterisk (*), and percent symbol (%). The [SHIFT] key must be held down while pressing a key to obtain the shifted character of that key. — In the Lower-case mode, pressing the [SHIFT] key while pressing any letter key will display the upper-case (capital) character of that letter.
SPACE BAR	• Operates just like the space bar on a regular typewriter. When you press the [SPACE BAR], the computer leaves a blank space between words, letters, or numbers. — The [SPACE BAR] can be used to correct errors. This will be explained in the section on Editing.

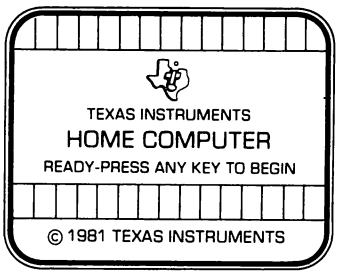
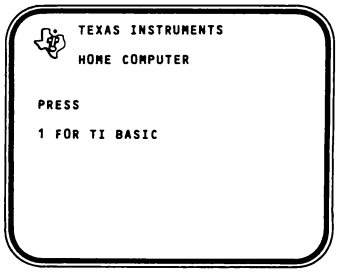
KEY	FUNCTION
CTRL	• Stands for "CONTROL." This key is used primarily for telecommunications to permit entry of special control characters. To enter a control character, hold down the CTRL key and press the appropriate letter or number key. (For a list of Standard ASCII control characters, refer to the Appendix of the *User's Reference Guide.*)
FCTN	• Stands for "FUNCTION." Several keys have an additional function that is obtained by holding down the FCTN key while the other keys are pressed. — Most of the functions — "AID," "CLEAR," "QUIT," etc. — are identified on the strip overlay packed with the computer. (If you have not inserted this strip into the slot above the top row of keys on the keyboard, do it now!) — Note that certain symbols (⇧, ~, [,], ?, ", etc.) are printed on the front of several letter keys.
FCTN 1 (DELete)	• Holding down the FCTN key while pressing the 1 key will permit you to delete characters from the program line. To delete characters, position the cursor (using the ⇦ or ⇨ key) over the character you wish to delete, then press the FCTN and 1 keys.
FCTN 2 (INSert)	• Puts the computer in Insert mode and permits you to insert characters into the middle of a program line. To insert characters, position the cursor (using the ⇦ or ⇨ key) over the character immediately to the right of the place you wish to insert characters, then press the FCTN and 2 keys (the computer is now in Insert mode). After you have pressed these keys, each time you type a character, the cursor and every character of the program line that is not to the left of the cursor are moved to the right. — Each character you type is inserted into the blank position left by the shifting of the cursor and other characters. — Note that characters shifted off the end of the program line are deleted from the line. — To get out of Insert mode, press the ENTER key.
FCTN 3 (ERASE)	• Erases the entire program line, including the line number you are retyping, *if you have not pressed* the ENTER key. — If the computer is in Edit mode, however, the entire line displayed for editing is erased *except the line number.*

KEY	FUNCTION
FCTN 4 (CLEAR or BREAK)	• When you press the FCTN 4 keys, you can perform several functions, depending on when you press these keys. — If a program is running, pressing FCTN 4 stops the execution of the program. The message BREAKPOINT AT LINE NUMBER is displayed. To continue execution, type CONTINUE and then press ENTER. — If you are typing in a program line but do not wish to enter it, pressing the FCTN 4 keys causes the program line to scroll (move) up on the screen, but that line is not entered (because you have not pressed the ENTER key yet). — If the computer is in Edit mode, the current line scrolls up on the screen and the computer leaves Edit mode. Any changes you made on the line before pressing the FCTN 4 keys are ignored. That is, the existing program line does not change.
FCTN 5 (BEGIN) FCTN 6 (PROC'D) FCTN 7 (AID) FCTN 8 (REDO) FCTN 9 (BACK)	• These keys (BEGIN, PROC'D, AID, REDO, and BACK) have special functions with software applications and will not be covered in this book. (Applications packages or software packages are programs that have been written for the user. These packages include games, mathematics courseware, personal record keeping, and the like.
FCTN D (⇨) or (Forwardspace)	• ⇨ or cursor right key moves the cursor to the right one character position. The cursor does not erase or change the characters on the screen as it passes over them. — When the cursor reaches the end of a line on the screen, it wraps around the screen and moves to the beginning of the next line down. (Note! When the cursor reaches the end of its "logical line," which is four screen lines, pressing the ⇨ has no effect.)

KEY	FUNCTION
FCTN S (⇦) or (Backspace)	• ⇦ or cursor left moves the cursor to the left one character position. Using the ⇦ does not delete or change the character it passes over. — When the cursor reaches the end of a line, it wraps around and moves up one row and to the extreme right-hand end of the row. (Note! If the cursor reaches the beginning of the line, pressing the ⇦ key has no effect.)
FCTN E (⇧)	• The up-arrow (⇧) and down-arrow (⇩) keys work exactly like the ENTER key, except in Edit mode. (The Edit mode will be explained on subsequent pages.)
FCTN X (⇩)	
FCTN = (QUIT)	• You press these keys any time you want the computer to return to the master screen, i.e., the screen you see when the computer is first turned on. — When you press these keys, all data or program information you entered will be erased.

REVIEW

TI-99/4A Power-Up Rules

YOUR ACTION	DISPLAY
1. Make certain the TI-99/4A micro-computer is connected properly (refer to the *User's Reference Guide* if you have questions).	
2. If the tape recorder is connected, it should be in the *STOP* mode. (This procedure assumes that you are not using a disk.)	
3. Turn on the video display and set the RF modulator to "Modulator." (Make certain that channel selection on the television knob matches that on the computer.)	
4. Turn on the TI-99/4A. The power switch is located on the lower-right front of the console.	
5. After a few seconds the message should appear on the screen as shown.	
6. Press any key on the keyboard and you will see:	
7. Now press [1] on the keyboard to select TI BASIC. The screen will appear as shown and you are ready to use TI BASIC.	TI BASIC READY >■ ←(cursor)

Note:

○ Whenever the flashing cursor (■) appears on the screen, the computer is waiting for you to enter something from the keyboard. The prompt character (>) marks the beginning of each line.

3.1
Getting It Together

- Step 1 — Write Your Program
- Step 2 — Get the Computer Ready
- Step 3 — Enter Your BASIC Program
- Step 4 — RUN Your Program
- Step 5 — Sign Off

Note:

○ Step 5 means turning the computer off when you have finished programming.

3.2
Typical Display Readout

```
1Ø          PRINT          "HELLO THERE"

2Ø          PRINT          "YOUR NAME"

3Ø          END

RUN
```

3.3

Writing Your First Computer Program

YOUR ACTION	DISPLAY
1. Before you start typing your program, always type NEW and press the [ENTER] key.	>TI BASIC READY >■
2. Type the line exactly as shown: 3. Use the [FCTN] key to type the quotation marks (") and the [SHIFT] key for the exclamation point (!). 4. Do *not* press the [ENTER] key yet!	>1Ø PRINT "HELLO THERE NAME!" ■ Ⓐ Ⓑ
5. Go back and examine your typed line *very carefully.* Did you make a mistake? If you did, just press the backspace key [⇦] ([FCTN][S]) to move the cursor back to that point and then type the correct character(s).	
6. Is everything OK? If it is, you can press [ENTER]. (This tells the computer to "look at" what you just typed in).	>1Ø PRINT "HELLO THERE NAME!" Ⓒ >■
7. The prompt > and cursor ■ should appear. The computer is saying, "It's your turn...I'm waiting for you."	

Note:

○ NEW is a command that erases any program that may have been in the computer's memory, and typing NEW is an important first step in programming.

Ⓐ Insert student's name.

Ⓑ A slash is used in this book to distinguish a zero (Ø) from the letter "O." Your screen, however, will not display the slash.

Ⓒ The TI-99/4A has 28 columns across the screen. If the line you typed has more than 28 characters (including spaces, numbers, and symbols), the characters will "spill over" or "wrap around" to the next line automatically.

3.4
Executing Your Program

YOUR ACTION

1. Tell the computer to execute or run your program. The command for this is simple: RUN.
2. So type RUN and press [ENTER].
3. If you made no mistakes, the display will read:
4. If it did not work, try again (i.e., check your program for errors).
5. If it did work, let out a yell, "HEY, I CAN DO IT TOO!"

   Go to next page (if you completed this one OK).

DISPLAY

```
>RUN
 HELLO THERE NAME!

 ** DONE **

>■
```

3.5
Common Errors

- Missing quotes (")
- Too many quotes
- Forgot the key word PRINT
- Forgot the line number
- Forgot to press ENTER
- Used the character "O" for the number "zero" (Ø).

Note:

○ A slash is used to help you recognize a zero. Look at your keyboard closely. The computer screen, however, displays the letter "O" with squared corners and a zero with rounded corners. Make certain you can distinguish between them!

3.6

Writing Your First Computer Program — Almost? (Correcting Errors)

PROBLEMS: (You forgot to follow instructions.)

1. **Missing Quotes (")**
   — You forgot to enclose everything after the word PRINT in quotation marks. Don't forget to use quotation marks if you want something printed!
2. **Too Many Quotation Marks**
   — You typed too many. That won't work either!
3. **Forgot the Key Word PRINT**
   — You forgot to type PRINT. How will the computer know to PRINT something if you don't tell it to?
4. **Forgot to Type the Line Number**
   — Line numbers tell the computer where to start. The computer always starts executing from the lowest numbered line unless you tell it to start elsewhere.

SOLUTION:

If you have already pressed ENTER, you must retype the entire line to correct your error.

- Type in the same line number you wish to change.
- Retype the line exactly as shown on the previous page. (But this time, be more careful!)
- Then, check the line over for errors.
- If everything is OK, don't forget to press ENTER! Pressing ENTER tells the computer to "look at" what you just typed and act accordingly.

Read this page if you had any errors! Then correct your errors before going to the next page.

3.7
Expanding Your Program

YOUR ACTION

1. You now have a program in the computer (unless you turned it off. If you did, retype the line as shown):
2. Type in Line 2Ø *exactly* as shown:
3. Check your new Line (2Ø) *very carefully,* especially the quotation marks.
4. Everything OK? Press [ENTER]. (Remember, always press [ENTER] if you want the computer to look at what you typed.)
5. Let's run your program. Type RUN and press [ENTER].
6. If you did it right, the screen will read:
7. If it did not work, check your program for errors.

Go to next page.

DISPLAY

```
>1Ø PRINT "HELLO THERE NAME!"
>2Ø PRINT "I'M GOING TO MAKE
  YOU A SUPERSTAR!"
■
```

```
>RUN
  HELLO THERE NAME!
  I'M GOING TO MAKE YOU A SUPE
  RSTAR!

  ** DONE **

■
```

3.8

Using the PRINT Statement for Spacing

YOUR ACTION	DISPLAY
1. Look at your video display. Would you like more space between the two lines? OK, this is how you do it.	HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPE RSTAR!
2. Type in a new line as shown and then press [ENTER].	>15 PRINT >■
3. Now type RUN and press [ENTER].	>RUN HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPE RSTAR! ** DONE ** >■
4. Wow! A PRINT "nothing" puts a space between what you told the computer to print in Lines 1Ø and 2Ø.	
5. Observe that the PRINT statement (Line 15) was placed between Lines 1Ø and 2Ø. Since you were smart enough to number your lines by 1Ø's, it was much easier to modify your program. (That's because you left room to insert new lines between the old ones.) Although it is perfectly legal to number program lines more closely (like 1, 2, 3, 4), don't do it.	

Go to next page.

3.9

Inserting Remarks into a Program (But Not Printing Them Out)

YOUR ACTION	DISPLAY
1. Type Line 5 and press [ENTER] .	>5 REM THIS IS MY FIRST COMPU TER PROGRAM
2. Type RUN and press [ENTER] . (It is the same output as before!)	>RUN HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPE RSTAR! ** DONE ** >■
Go to next page.	

Note:

- ○ REM stands for REMark. It is an important statement in BASIC.
- ○ It is often convenient to insert REMarks into a program. The main reason for inserting REMarks is so that you or someone else can refer to them later and know what the program is for and how it is used.
- ○ When you tell the computer to execute the program by typing RUN and pressing [ENTER] , it will skip over any number line that begins with the statement REM. The REM statement will have no effect on the output of the program!

3.10

Listing Your Program (Looking at Your Program to See What It Contains)

YOUR ACTION

1. Type CALL CLEAR and press [ENTER].
2. Type LIST and press [ENTER]: (You can LIST your program anytime the prompt > appears on the screen.

DISPLAY

```
>LIST
 5 REM  THIS IS MY FIRST COMP
 UTER PROGRAM
 10 PRINT "HELLO THERE NAME!"
 15 PRINT
 20 PRINT "I'M GOING TO MAKE
 YOU A SUPERSTAR!"
>■
```

3. You might want to LIST only one line. Type LIST 20 and press [ENTER]. (Make sure there is a space between the word LIST and the line number, otherwise you will get an INCORRECT STATEMENT error.)

```
>LIST 20
 20 PRINT "I'M GOING TO MAKE
 YOU A SUPERSTAR!"
>■
```

4. You might also want to LIST several program lines, starting at one line and ending at another. For example, type LIST 10 - 20 and press [ENTER].

Go to next page.

```
>LIST 10-20
 10 PRINT "HELLO THERE NAME!"
 15 PRINT
 20 PRINT "I'M GOING TO MAKE
 YOU A SUPERSTAR!"
>■
```

Note:

○ CALL CLEAR is the command that will clear the screen. It's a good idea to do this from time to time to make it easier to read.

3.11

Ending Your Program

YOUR ACTION

1. The end of a program is the last statement you want the computer to execute. Most computers require you to place an END statement after this point, so that the computer will know it is finished. However, the TI-99/4A does *not* require an END statement.
2. Let's add an END statement to your program. Type and [ENTER]:
3. Now type RUN and press [ENTER].
4. No change from before! The program ended, but it did not print "END."
5. Let's make it print "THE END." (How do we do that?)
6. Oh, I remember! We need a PRINT statement. So let's try it. Type and [ENTER]:
7. Now RUN your program.
8. It worked again! (If not, check the program.)
9. Note that there is no space between "THE END" and the line above it. Why? Because you did not tell the computer to put a space between them!

DISPLAY

```
>99 END
>RUN
 HELLO THERE NAME!

 I'M GOING TO MAKE YOU A SUPE
 RSTAR!"

 ** DONE **

>■
```

```
>98 PRINT "THE END"
>RUN
 HELLO THERE NAME!

 I'M GOING TO MAKE YOU A SUPE
 RSTAR!
 THE END

 ** DONE **

>■
```

3.12

Using the Left-Arrow (⇦) Key to Save Retype Time (Before You Press the ENTER Key)

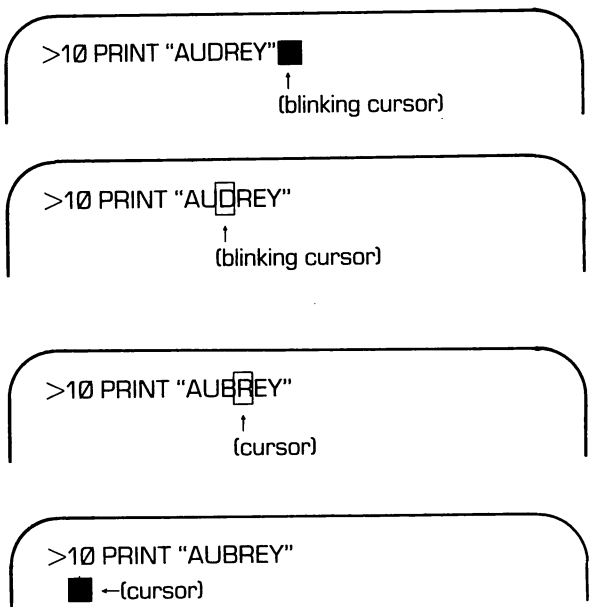
YOUR ACTION

1. You typed Line 1Ø as shown but have *not* pressed ENTER (blinking cursor at the end of that line indicates you have not pressed ENTER).
2. You wish to change the "D" to a "B" or to PRINT AUBREY. So you use the ⇦ key to move the cursor to the left one space at a time. (Don't forget to use the FCTN key.)
3. Now type "B" but *don't* press ENTER yet. (Note that the cursor has moved to the next letter "R.")
4. If you have finished typing the line and everything is correct, press ENTER. (Note that after you press ENTER the blinking cursor moved to the beginning of the next line.)
5. Remember you can always retype the entire line but the ⇦ key saves you time.

Note:

○ Type NEW before you start.

DISPLAY



3.13
Some Tips on Editing

If you made a mistake in typing in a program line or if you wish to change an existing program line, you can change a line by entering the Edit mode.

• Editing One Line Only

To enter Edit mode, you type the EDIT command followed by a line number (e.g., EDIT 1Ø and then press the [ENTER] key).

— When you enter Edit mode, the program line you requested is displayed on the screen. The prompt character (>) is not displayed to the left of the line when you are in the Edit mode.

— When the requested line is displayed, the flashing cursor is positioned in the second character position to the right of the line number (see example on next page).

— Changes can be made to any character on the line except the line number.

• Editing Several Lines

When you press the [ENTER] key, all changes you have made to the program line are entered into the computer's memory and the computer leaves the Edit mode.

— If you want the computer to stay in Edit mode because you want to make changes to other program lines, then you should press [FCTN] and [E] ([⇧]) or [FCTN] and [X] ([⇩]).

— Pressing the [FCTN] and [E] ([⇧]) keys causes all changes you have made to the program line to be entered into the computer's memory. The next lower-numbered line in the program is then displayed for editing. If no lower program line exists, then the computer leaves Edit mode. (Note that the cursor does not have to be at the end of the line for the entire line to be entered by the [⇧] key.)

— Pressing the [FCTN] and [X] ([⇩]) keys causes all changes you made to the program line to be entered into the computer's memory. The next higher-numbered program line is then displayed by editing. If no higher program line exists, then the computer leaves the Edit mode. The cursor does not have to be at the end of the line for the entire line to be entered by the [⇩] key.)

3.14

Using the Edit Mode to Correct Errors (After You Have Pressed the ENTER Key)

YOUR ACTION

1. Type NEW and press ENTER.
2. Type in Line 10 exactly as shown and press ENTER .
3. Since you have entered this line in the computer, you must enter the Edit mode by typing:
   (Leave a space between the word "EDIT" and "10".)
4. Now press ENTER . (Note the position of the cursor.)
5. Move the cursor over the "I" in "IRROR" by using the ⇨ key. (Don't forget to hold down the FCTN key.)
6. Change the "I" to "E" by typing "E." (Do not press the ENTER key yet!)
7. If this were the only change you wanted to make, you would merely press the ENTER key to get out of Edit mode. But let's change the word "IS" to "WAS."
   a. Using the ⇦ key, move the cursor over the "I" in the word "IS."
   b. Now type "A." (Note the position of the cursor.)

DISPLAY

```
>TI BASIC READY
>10 PRINT "THIS IS AN IRROR"
```

```
>EDIT 10
```

```
>EDIT 10
 10 PRINT "THIS IS AN IRROR"
      ↑
 (cursor)
```

```
>EDIT 10
 10 PRINT "THIS IS AN IRROR"
                      ↑
                 (cursor)
```

```
>EDIT 10
 10 PRINT "THIS IS AN ERROR"
                      ↑
                 (cursor)
```

```
>EDIT 10
 10 PRINT "THIS IS AN ERROR"
                ↑
           (cursor)
```

```
>EDIT 10
 10 PRINT "THIS AS AN ERROR"
                ↑
           (cursor)
```

3.14 (Cont.)

YOUR ACTION	DISPLAY
c. To insert a "W," position the cursor over the "A" using the [⇦] key. (Don't forget to hold down the [FCTN] key!)	>EDIT 10 10 PRINT "THIS [A]S AN ERROR" ↑ (cursor)
d. Press the [INSERT] key. (i.e., hold down [FCTN] and press [2].)	>EDIT 10 10 PRINT "THIS [A]S AN ERROR" ↑ (cursor still here)
e. Now type "W" and you will see that a "W" was inserted. (Notice that the cursor moved to the right.)	>EDIT 10 10 PRINT "THIS W[A]S AN ERROR" ↑ (cursor)
f. If you have finished making all of your changes, press the [ENTER] key.	>EDIT 10 10 PRINT "THIS WAS AN ERROR" >■

3.15

Some Helpful Keys and Commands to Remember

ACTION	KEY(S) TO PRESS	COMMAND
• Clear screen and home cursor	[C][A][L][L] (space bar) [C][L][E][A][R]	CALL CLEAR
• Enter data	[ENTER], [FCTN][⇧], [FCTN][⇩]	———
• Execute a program	[R][U][N] and [ENTER]	RUN
• Stop program execution	[FCTN] [4] ([CLEAR])	BREAK STOP END
• CONTINUE program	[C][O][N][T][I][N][U][E] and [ENTER]	CONTINUE
• LIST program	[L][I][S][T] and [ENTER]	LIST
• Backspace (cursor left)	[FCTN] [⇦]	———
• Forward space (cursor right)	[FCTN] [⇨]	———
• Return to master screen	[FCTN][=] ([QUIT])	——— Ⓐ
• Erase an entire program line (before [ENTER] key is pressed	[FCTN] [3] ([ERASE])	———

Note:

Ⓐ When you press [QUIT], all data or program material you have entered will be erased.

3.16

Learned in This Section

COMMANDS*	KEY WORDS**	MISCELLANEOUS	SPECIAL FUNCTION KEYS
• CALL CLEAR • LIST — LIST MM • NEW • RUN	PRINT "MESSAGE" PRINT (SPACE) REM END	> Prompt ■ Cursor " " Quotation Marks Line Numbering Keyboard Layout TI-99/4A Power-Up Rules QUIT DONE	FCTN ENTER ⇦ ⇨ ALPHA LOCK
*Executed as soon as you type them and press ENTER	**Used to make statements. Statements are executed after you type RUN and press ENTER		

Note:

- ◯ MM = Any line number (e.g., 1Ø, 2Ø, 3Ø, etc.)
- ◯ Always leave a space between a key word and a line number (MM); otherwise an error will occur.
- ◯ You will use the computer with the ALPHA/LOCK key down (Upper-case mode) for all lessons in this book.
- ◯ If you don't understand everything on this page, stop! Go back over this section until you understand it thoroughly!

ASSIGNMENT 3-1

1. Write a program to print on separate lines:
   a. Your name
   b. Your entire address
   c. Your telephone number
2. Expand your program to include the following:
   a. REM statements to describe your program
   b. Spacing between each of the lines displayed (printed)
   c. An END statement
3. Type your program and enter it.
4. RUN your program.
5. LIST your program.

Note:

○ Write your program on paper and get it checked by your teacher first.

PRACTICE 3

Writing and Running Your First Program

1. Write a program to PRINT the following:
   a. Your name (first and last)
   b. Your school's name
   c. Your teacher's name
2. Enter and RUN it.

PRACTICE 4

Inserting Remarks and Spacing into Your Program

1. If you have erased the program from Practice 3, rewrite the program and do the following: (If you still have the program from Practice 3 in the computer, you do not have to rewrite the program.)
   a. Add a new program line with a REM statement to your program (any remarks you want to make).
   b. Have the computer insert one space between your name and your school's name in the output on the display (that is, you add the necessary program line).
   c. Have the computer insert two spaces between your school's name and your teacher's name in the output on the display.

PRACTICE 5

Listing and Ending Your Program

1. Rewrite the program from Practice 4 and do the following (Again, if you have the program in the computer, you don't have to rewrite it. But in case you don't know what is in the computer, just type NEW and rewrite the program.):
   a. Add an END statement to tell the computer it is the end of your program.
   b. Add a statement to have your computer PRINT "THE END."
   c. RUN your program.
2. LIST your program.
   a. How large is your program now? (How many lines?)
   b. Copy the program in your notebook.

PART 4

More Programming Tools

What You Will Learn

1. To enter and RUN more BASIC programs: mathematical programs, area of rectangle program.
2. To explain the order of mathematical operations using the M.D.A.S. rule.
3. To explain the purpose and use of the key word: **LET**.
4. To explain the purpose and use of the mathematic operators: multiply (*), divide(/), add (+), subtract (—), exponentiate or raise a number to a power(^).
5. To explain the function and use of commas, semicolons, and PRINT zones.
6. To identify variables that can be used with TI BASIC.

REVIEW

COMMANDS*	KEY WORDS**	MISCELLANEOUS	SPECIAL FUNCTION KEYS
• CALL CLEAR • LIST — LIST MM • NEW • RUN	PRINT "MESSAGE" PRINT (SPACE) REM END	> Prompt ■ Cursor " " Quotation Marks Line Numbering Keyboard Layout TI-99/4A Power-Up Rules QUIT DONE	FCTN ENTER ⇦ ⇨ ALPHA LOCK
*Executed as soon as you type them and press ENTER	**Used to make statements. Statements are executed after you type RUN and press ENTER		

Note:

- ◯ MM = Any line number (e.g., 1Ø, 2Ø, 3Ø, etc.)
- ◯ Always leave a space between a key word and a line number (MM); otherwise an error will occur.
- ◯ You will use the computer with the ALPHA/LOCK key down (Upper-case mode) for all lessons in this book.
- ◯ If you don't understand everything on this page, stop! Go back over this section until you understand it thoroughly!

4.1

Math Operators

= (Equals)	* (Multiply)
+ (Add)	/ (Divide)
− (Subtract)	^ (Exponentiation)

Note:

○ Exponentiation (^) means raising a number to a power like 2^2, 2^3, or 2^4.

4.2

Order of Arithmetic Operations

- Multiply → Divide → Add → Subtract (left to right)
  — "My Dear Aunt Sally"
- If parentheses are used:
  — the innermost set of parentheses is simplified first, followed by each successive set outward.
  — the M.D.A.S. order is followed inside all sets of parentheses.

4.3

Order of Operations Example (without Parentheses)

- If there are no parentheses, the computer performs operations by going from left to right doing exponentiation operations (Λ) first. Then (*) and (/) are done in order from left to right and finally (+) and (—) are done in order from left to right. (Remember M.D.A.S.!)
- Example:

```
4 + 5 * 4 Λ 3 — 4/2 =
4 + 5 *[64]  — 4/2 =
4 +  [320]   — 4/2 =
4 +   320    — [2] =
[324]        —  2  =[322]
```

4.4

Order of Operations Example (with Parentheses)

- If there are parentheses, the computer starts at the inner pair of parentheses and converts everything to a single number. Then the computer repeats the process with the next pair of parentheses working "inside" out.
- Example:

  ((6 + 4) * 2) /4 =
  ([10] * 2) /4 =
  [20] /4 = [5]

Note:

○ The same answer can be worked out first on paper and then on the computer. If you type in

PRINT ((6+4) * 2) /4

and press [ENTER], the computer will print out 5.

EXERCISE 4-1

- You try some now (without parentheses).
  1. $2 \wedge 3 + 4 * 5 - 4/2 * 5 =$ _____
  2. $14 - 2 * 2 + 6 - 2 * 3 * 2 =$ _____
  3. $14/2 * 3 - 2 \wedge 3 + 4 =$ _____
- Now try some with parentheses.
  1. $6 + (9 * 2) =$ _____
  2. $(6 + (9 * 2)) * 5 =$ _____
  3. $3 * ((4 + (6 * 2)) * (9/3 - 1)) =$ _____

Note:

○ You should work these out on paper. If you use the computer to check the answers, be sure to use the word PRINT and leave off the equals sign (=). For example, type

PRINT 2 ∧ 3 + 4 * 5 − 4/2 * 5

and press ENTER.

SUMMARY

Tips on Using Parentheses

- When in doubt, use parentheses. They can't do any harm!
  — Use parentheses around operations you want performed first.
- Make sure that every left parenthesis has a matching right parenthesis.
  — Count them to be sure!
- Order of operations:
  — Innermost pair of parentheses first (M.D.A.S. rule inside parentheses).
  — Then work "inside" out.
  — In case of a "tie," computer starts to the left and works right doing exponentiation (^) and the M.D.A.S. rule.

4.5

Numeric Variable Names Used with TI BASIC

- May be one or more characters in length
  — up to 15 characters may be used
  — must not be a reserved word (see below)
- Must begin with a letter (A–Z), an at sign (@), a left bracket ([), a right bracket (]), a back slash (\), or a line (_)
  — may be followed by another letter, the at sign (@), or the line (_)
  or
  — may be followed by a digit (Ø–9)
- Some examples of numeric variable names include:
  — A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.
  — A1, A2, B1, B2, C3, C5, D9, N9, P4, Q1, R6, Y7
  — AA, AZ, GP, MU, ZZ, BB, XY, LL, FG, LE, RE
  — ALPHA, BASE, TOTAL, SUM, TABLE, NAME, DATE
  (You get the picture! Using the above combinations, you can use approximately 9ØØ variable names.)
- There are some words with special meaning in the BASIC language and they *cannot* be used as numeric variable names.
  — The complete list of reserved words, which cannot be used in variable names, appears in the *TI User's Reference Guide.*

4.6

Program for a Mathematical Operation

Line No.	Key Word	Other Part of Statement	
1Ø	LET	X = 5	ENTER
2Ø	LET	Y = 12	ENTER
3Ø	LET	Z = X*Y	ENTER
4Ø	PRINT	Z	ENTER
99	END		
RUN			

Note:

- LET is an optional key word for TI BASIC. Some computers require you to use LET, however. Beware of this if you use another computer.
- ENTER is not part of the program. It is just a reminder to press it after each line.

4.7

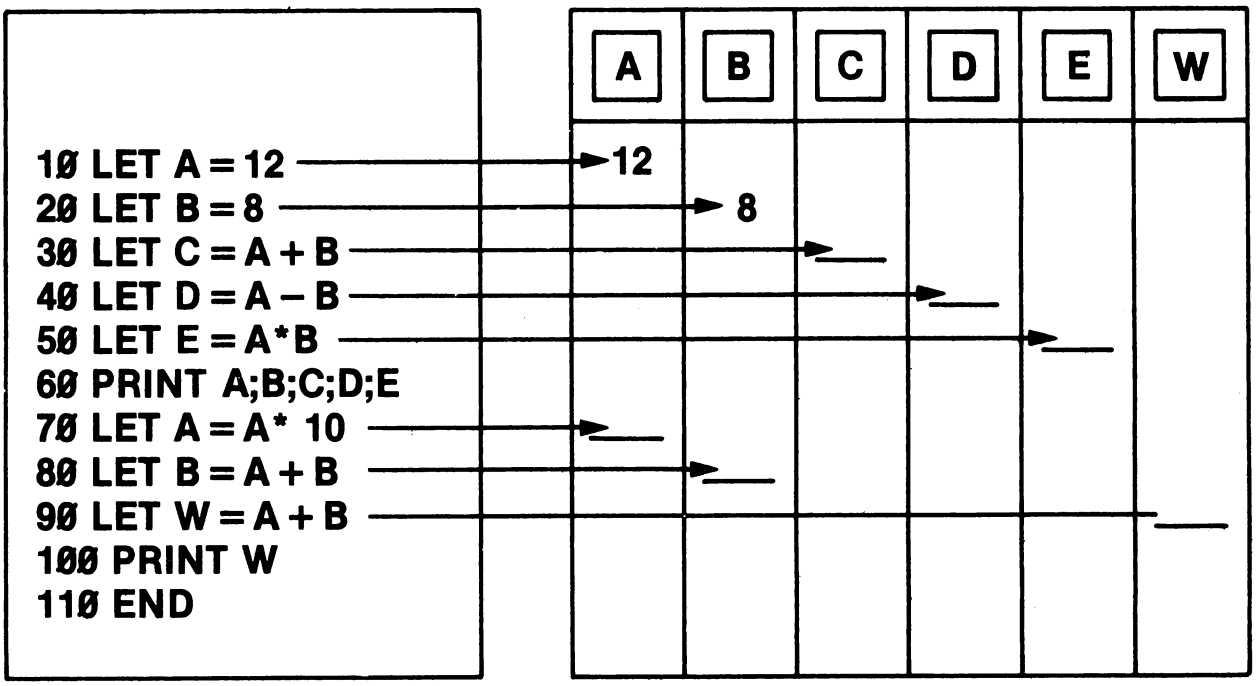
Analysis of the Program for a Mathematical Operation

Line No.	Statement	Meaning to Computer
1Ø	LET X = 5	Assign a value of 5 to variable X
2Ø	LET Y = 12	Assign a value to 12 to variable Y
3Ø	LET Z = X*Y	Take the values of X and Y, multiply them together, and assign the resulting value to the variable Z
4Ø	PRINT Z	PRINT the value of Z (which is 6Ø in the example)
99	END	END Program
RUN		Execute Program

EXERCISE 4-2

Assigning Numeric Values to Variables

- Fill in the values of the variables on each line in the program.



4.8

A Mathematical Program: Area of Rectangle

YOUR ACTION	DISPLAY
1. Type NEW. 2. Type and enter.	TI BASIC READY >10 REM AREA OF A RECTANGLE >20 REM AREA = LENGTH X WIDTH >30 LET L = 10 >40 LET W = 5 >50 LET A = L*W >60 PRINT A
3. Type RUN and press [ENTER].	>RUN 50 ** DONE ** >■

Note:

○ We said in Line 60 PRINT A. There were no quotes around the letter A because we wanted the computer to PRINT the *value* of A. If we wanted the computer to PRINT the exact word or letter, we would put quotes around the word or variable.

4.9

Area of Rectangle Program Modified

YOUR ACTION

1. Add Line 70.
2. Type RUN.

DISPLAY

```
>70 PRINT"THE AREA =";A

>RUN
  50
 THE AREA = 50
```

Ⓐ

3. Add Line 80.
4. Type RUN.

```
>80 PRINT"THE AREA IS",A

>RUN
  50
 THE AREA = 50
 THE AREA IS        50
```

Ⓑ

5. Add Lie 90.
6. Type RUN.

```
>90 PRINT"THE AREA IS";A;"SQ.
 INCHES"

>RUN
  50
 THE AREA = 50
 THE AREA IS        50
 THE AREA IS 50 SQ.INCHES

 ** DONE **
```

Ⓒ
Ⓓ

7. LIST the program.

```
>LIST
 10 REM    AREA OF A RECTANGLE
 20 REM    AREA = LENGTH * WIDT
 H
 30 LET L=10
 40 LET W=5
 50 LET A=L*W
 60 PRINT A
 70 PRINT "THE AREA =";A
 80 PRINT "THE AREA IS",A
 90 PRINT "THE AREA IS";A;"SQ
 . INCHES"
>■
```

Note:

Ⓐ The semicolon in Line 70 caused the value of A to be PRINTed next to a label, namely, THE AREA =.

Ⓑ The comma in Line 8Ø caused the value of A to be PRINTed separated from a label.

Ⓒ The semicolons in Line 9Ø caused the value of A to be PRINTed between two labels. A space is automatically inserted before and after a numeric variable when PRINTed. This may not be true for other microcomputers.

Ⓓ There are four different outputs in this RUN since the program contained four different PRINT statements (Lines 6Ø, 7Ø, 8Ø, and 9Ø).

ASSIGNMENT 4-1

1. Write a program to find the area of a triangle.
   a. GIVEN: A = 1/2BH where B = 5, H = 1Ø
   b. Include REM statements
   c. Have the program PRINT "THE AREA="; (your answer); "SQ. FT."
2. Write a program to find the volume of a rectangular solid.
   a. GIVEN: V = LWH where L = 5, W = 1Ø, H = 2
   b. Include REM statements
   c. Have the program PRINT "THE VOLUME="; (your answer); "CUBIC IN."
3. Write a program to convert Fahrenheit to Celsius.
   a. GIVEN: C = (F − 32) × (5/9) where F = 75°
   b. Change the value of F to 45° and RUN the program again.
4. Write a program to convert Celsius to Fahrenheit.
   a. GIVEN: F = 9/5 × C + 32 where C = 2Ø°
   b. Change the value of C to 35° and RUN the program again.

Note:

◯ Remember that a formula must be written in proper computer form in your programs, i.e., A = 1/2*B*H for area of a triangle.

SUMMARY

PRINT and LET

- LET is an optional key word in TI BASIC.
  — Other computers using BASIC might require the use of LET.
- A comma in a PRINT statement tells the computer to leave several spaces between items separated by the commas.
- A semicolon inserts one space between two items it is separating on the same line if the two items include a numeric variable and a "message."

Note:

○ 1Ø PRINT "A" tells the computer to print the letter A.
1Ø PRINT A tells the computer to print the value of the variable A (a number).

4.10
PRINT Zones

Zone 1 14 Spaces	Zone 2 14 Spaces
**LEEDS*MIDDLE	SCHOOLCOMPUTER

- The TI-99/4A output is divided into two PRINT zones.
  — Each PRINT zone has 14 spaces for up to 14 characters.
  — The TI-99/4A can display up to 28 characters per line ($2 \times 14 = 28$).
- Commas are used to tell the computer to move to the next PRINT zone.
  — The cursor moves to the next PRINT zone each time a comma is encountered.

Note:

○ Try typing in the words shown above. Count each character or symbol as you type it. Also note that there are no spaces between characters.

4.11

PRINT Zones and the Use of Commas

YOUR ACTION	DISPLAY
1. Type NEW and press [ENTER].	TI BASIC READY
2. Type Line 10 and press [ENTER].	>10 PRINT"ZONE 1","ZONE 2"
3. Type RUN and press [ENTER].	>RUN ZONE 1 ZONE 2 Ⓐ ** DONE ** Ⓑ

Note:

Ⓐ There are two (2) 14-character PRINT zones per line (since 2 × 14 = 28, the screen can display up to 28 characters per line).

Ⓑ The comma tells the computer to move to the next PRINT zone each time a comma is encountered in a PRINT statement.

4.12

Semicolon versus Comma

YOUR ACTION

1. Type NEW and press [ENTER] .

2. Type Line 1Ø exactly as shown, then press [ENTER] .

3. Type Line 2Ø exactly as shown, then press [ENTER] .

4. Type RUN and press [ENTER] .

5. Type NEW and press [ENTER] .

6. Type Lines 3Ø, 4Ø, 5Ø, and 6Ø as shown.

7. Type RUN and press [ENTER] .

DISPLAY

```
 TI BASIC READY

>1Ø PRINT"A";"SEMICOLON";"PAC
 KS";"ITEMS";"CLOSE";"TOGETHE
 R"

>2Ø PRINT"BUT A","COMMA","LEA
 VES","SPACES"
```

```
>RUN
 ASEMICOLONPACKSITEMSCLOSE
 TOGETHER
 BUT A              COMMA
 LEAVES             SPACES
```

```
 TI BASIC READY

>3Ø LET A=5
>4Ø LET B=1Ø
>5Ø LET C=15
>6Ø PRINT A;B;C
```

```
>RUN
  5  1Ø  15

 ** DONE **
```

Note:

○ On the TI-99/4A, when the semicolon is used between two numeric *variables* in a PRINT statement, the computer automatically inserts one space between them. This might not be true with other computers, so beware!

SUMMARY

Use of the Semicolon and Comma

- The effect of the semicolon varies from computer to computer, but it is always true that a semicolon leaves less space within the output than a comma.
- GENERAL RULE:
  When you want more than one item on the same line and
  — if you want your results or output spread out, use a comma
  — if you want your results or output closer together, use a semicolon
- EXCEPTION:
  A numeric variable will have a space before and after the number when PRINTed on the TI-99/4A. For example, if your program contained the following line:

  60 PRINT"THE AREA IS";A;"SQ. INCHES"

  the output will look like this (if A = 50):

  THE AREA IS 50 SQ. INCHES

  Notice that there is space before and after the number 50 in the output, even though the PRINT statement contains semicolons.

PRACTICE 6
Perimeter of a Rectangle Program

Part I

1. Enter and RUN this program:

```
10 REM PERIMETER OF A RECTANGLE
20 REM P=2*L+2*W
30 LET L=9
40 LET W=4
50 LET P=2*L+2*W
60 PRINT P
```

2. Add a new program line to include a label on your answer. For example, THE PERIMETER OF THE RECTANGLE IS 26 INCHES.
3. Add new program lines to PRINT the following:
   a. THE LENGTH OF THE RECTANGLE IS 9 INCHES.
   b. THE WIDTH OF THE RECTANGLE IS 4 INCHES.

Part II

1. *Do not* type NEW.
2 Change the values of L and W in the program. (Think before you change the lines! How many lines do you have to change? Change only those lines!)

PRACTICE 7

Program Using Mathematical Operators

1. Enter and RUN this program:

```
10 REM MATH PROBLEMS
20 LET A=75
30 LET B=50
40 LET C=A+B
50 PRINT C
```

2. Change the values of A and B in the program and RUN it. Fill in the results: A=_____, B=_____, C=_____.
3. Add a program line to label the answer. Example: THE SUM IS (your answer).
4. Write a program to multiply (*) two numbers (any two).
5. Add the program line to PRINT: "THE PRODUCT OF" (your no.) "*" (your no.) "IS" (your answer). Example: THE PRODUCT OF 5*5 IS 25.
6. Write a program to divide (/) two numbers (any two).
7. Add the program line to PRINT: "THE QUOTIENT OF" (your #) "/" (your #) "IS" (your answer). Example: THE QUOTIENT OF 10/2 IS 5.
8. Write a program to subtract (—) two numbers (any two).
9. Add the program line to PRINT: "THE DIFFERENCE BETWEEN" (your #) "—" (your #) "IS" (your answer). Example: THE DIFFERENCE BETWEEN 10—5 IS 5.

Additional practices for this part will be found in the back of the book.

PART 5

Scientific Notation

What You Will Learn

To understand and use scientific notation.

Review and Feedback

The purpose of this part of the program is to evaluate students' overall performance and determine which students are having problems. The students who are having problems will be given the opportunity to review concepts they have not mastered. The review and feedback phase is divided into the following parts:

1. Exam — written/lab
2. Open discussion with students about their concerns and interests
3. Evaluation of student's performance
4. Recommendations

5.1
Scientific Notation

- Scientists often express large numbers like 186,ØØØ and small numbers like Ø.ØØØ15 as the product of two numbers.
- Example:

  a. 186,ØØØ $= 1.86 \times 1Ø^{5}$
  b. Ø.ØØØ15 $= 1.5 \times 1Ø^{-4}$
  c. 764,ØØØ $= 7.64 \times 1Ø^{5}$
  d. Ø.Ø347 $= 3.47 \times 1Ø^{-2}$
  e. 5,ØØØ,ØØØ $= 5 \times 1Ø^{6}$

5.2

Scientific Notation (Cont.)

Ordinary Notation		Scientific Notation		Scientific Notation on TI-99/4A
50,000,000,000	=	5×10^{10}	=	5.E+10
.00000000003	=	3×10^{-11}	=	3.E−11
.0000000000006	=	6×10^{-13}	=	6.E−13
1300000000000000	=	1.3×10^{15}	=	1.3E+15
1780000000000000	=	1.78×10^{16}	=	1.78E+16

- The TI-99/4A uses scientific notation for very large and very small numbers.
- Rule 1: E + 13 means move the decimal point 13 places to the right.
- Rule 2: E − 13 means move the decimal point 13 places to the left.

Note:

○ Numbers with 11 or more digits are automatically converted to scientific notation on the TI-99/4A.

ASSIGNMENT 5-1

1. Enter and RUN the following program:

```
5 CALL CLEAR
10 LET A = 5000000000000
20 LET B = .0000000000123
30 LET C = .00000000017865
40 LET D = 17650000000000
50 PRINT A,B,C,D
```

2. Experiment with scientific notation until you feel comfortable with it.

REVIEW AND FEEDBACK

1. Quiz — written/lab
2. Open discussion with students on concerns and interest
3. Evaluation of students' performance
4. Recommendations

Feedback Questionnaire

1. Do you like working with computers? _____ (yes, no) If not, why not? _____
   _____
2. What things do you like most about computers? _____
   _____
3. What do you dislike most about computers? _____
   _____
4. If you were a design engineer and could design the computer to do anything you wanted it to, what kinds of things would you include in your design? (Use your imagination!)
   _____
   _____
5. What was the hardest thing for you to understand about the computer so far? _____
6. What was the easiest thing for you to understand? _____
   _____
7. Were you afraid or nervous when you first used the computer? _____ (yes, no)
8. Do you feel comfortable using the computer now? _____ (yes, no)
9. Would you prefer to be doing something else rather than learning about computers? _____ (yes, no) If yes, what would you like to do? _____
   _____
10. Is the teacher going too fast, too slow, or just right for you? _____
11. Do you find the lessons interesting, boring, or so-so? _____
12. If you could teach this course, what would you do to make the lessons more interesting? _____
    _____
13. Have you decided what you want to do for a vocation? _____ (yes, no)
    If yes, what? _____
14. Would you like to take additional courses to learn more about computers and programming? _____ (yes, no)
15. Do you have any additional comments? _____
    _____
    _____
    _____
    _____

PRACTICE 8
Scientific Notation

1. Convert the following to standard scientific notation (Example: 5,ØØØ,ØØØ = 5×1Ø^{6}):

   a. 7,12Ø,ØØØ,ØØØ
   b. .ØØØØØ7
   c. .ØØØØØØØ8
   d. 6,1ØØ,ØØØ,ØØØ,ØØØ
   e. 8ØØØØØØØØØØØØØØØØ (16 zeroes)
   f. 8ØØØØØØØØØØØØØØØ (15 zeroes)
   g. 913,ØØØ,ØØØ,ØØØ
   h. 77,ØØØ,ØØØ,ØØØ,ØØØ
   i. 4Ø9,ØØØ,ØØØ,ØØØ,ØØØ
   j. 321ØØØØØØØ

2. Change the above numbers to computer scientific notation used in the TI-99/4A. (Example: 5ØØ,ØØØ,ØØØ,ØØØ = 5.E+11).

PART

Relational Operators and IF-THEN/GOTO Statements 6

What You Will Learn

1. How computers compare (or relate) one value with another.
2. To explain the purpose and use of the six relational operators: =, >, <, <=, >=, <>.
3. To explain the purpose and use of the key words **IF-THEN, GOTO.**
4. To write, enter, and RUN programs that use IF-THEN and GOTO statements.
5. To understand and use the counting program.

6.1
Relational Operators

- Relational operators allow a computer to compare one value with another.
  — The three relational operators include:

Symbol	Meaning	Examples
=	Equals	A = B
>	Greater than	A > B
<	Less than	A < B

— Combining the three operators above we have:

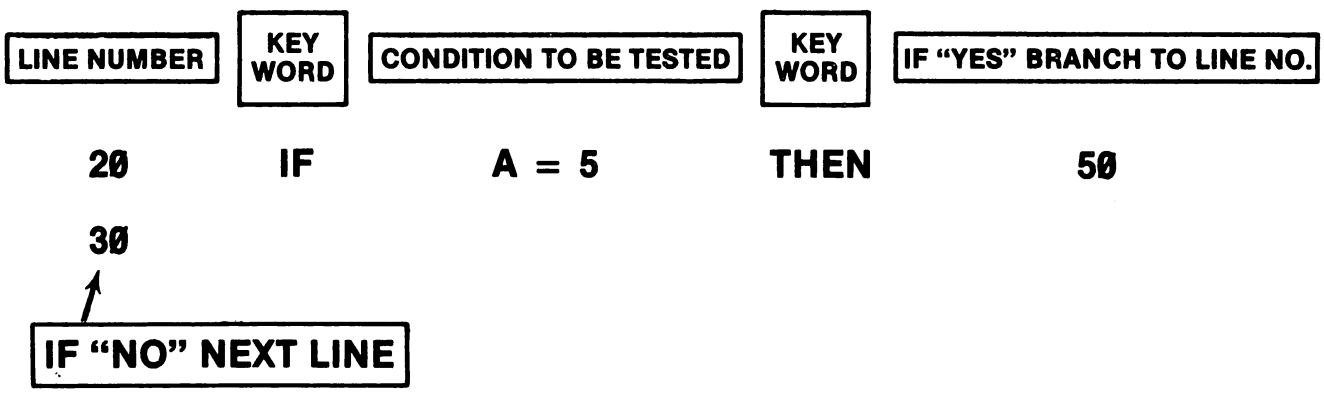
<>	Is not equal to	A <> B
<=	Less than or equal to	A <= B
>=	Greater than or equal to	A >= B

Note:

○ To distinguish between < and >, just remember that the smaller part of the < symbol points to the smaller of two quantities being compared.

6.2
IF-THEN (Conditional Branching)

- IF-THEN is used in conditional branching.
  — That is, the program will "branch" to another part of the program on the condition that it passes the test it contains.
  — If the test fails, the program simply continues to the next line.
- Example:



6.3
Sample Program Using IF-THEN

- Program:

```
10 LET A = 5
20 IF A = 5 THEN 50
30 PRINT "A DOES NOT EQUAL 5"
40 END
50 PRINT "A EQUALS 5"
RUN
```

- The screen should display:

```
A EQUALS 5
```

- Why is Line 20 a conditional branching statement?
  — What's the condition or test?

EXERCISE 6-1
IF-THEN

Given: A = 10, B = 20, C = 30

Exercise No.	Statement	Condition Is (T or F)	Branch to (Line N)	
1.	10 IF A = B THEN 40	F	20	Ⓐ
2.	10 IF A <> B THEN 50			
3.	10 IF A > B THEN 60			
4.	10 IF A < B THEN 70			
5.	10 IF C <= A + B THEN 80			
6.	10 IF C > A + B THEN 90			
7.	10 IF B > A THEN 100			
8.	10 IF B/A >= C/A THEN 110			
9.	10 IF A * B <= A * C THEN 120			
10.	10 IF C/A <= A * B THEN 130			

Note:

Ⓐ If condition is false (F), the computer will execute the next line (i.e., 20).

6.4

A Counting Program Using IF-THEN

YOUR ACTION	DISPLAY
1. Type in these lines.	>10 LET J=0 >20 LET J=J+1 >30 PRINT J >40 IF J < 10 THEN 20
2. RUN the program.	1 2 3 4 5 6 7 8 9 10

EXERCISE 6-2

Modify above program to count to 50 by 5's.

6.5

IF-THEN Counter Program

```
10 J = 0
20 J = J + 1
30 PRINT J,
40 IF J < 4 THEN 20
50 END
```

Program Analysis

Initialize	Program Execution	"J" Counter Status	Display
	10 J = 0	[0]	
1st Time	20 J = J + 1	[1] = 0 + 1	
	30 PRINT J,		
	40 IF J < 4 THEN 20		
2nd Time	20 J = J + 1	[2] = 1 + 1	
	30 PRINT J,		
	40 IF J < 4 THEN 20		
3rd Time	20 J = J + 1	[3] = 2 + 1	
	30 PRINT J,		
	40 IF J < 4 THEN 20		
4th Time	20 J = J + 1	[4] = 3 + 1	
	30 PRINT J		
	40 IF J < 4 THEN 20		
END	50 END		1 2 3 4

EXERCISE 6-3
GOTO (Unconditional Branching)

- Type and RUN this program:

```
10 CALL CLEAR
20 PRINT "YOUR NAME";
30 GOTO 20
```

- What happened?
  - — Do you know how to stop the program? (What about the [FCTN] and [CLEAR] keys?)
  - — What does Line 30 tell the computer to do?
  - — Are there any tests or conditions to be satisfied in Line 30 before it does what it has to do?
  - — Do you understand now why the GOTO statement is called an unconditional branching statement?

EXERCISE 6-4
GOTO/IF-THEN

- Study the program below and write the message that would be printed if the program were executed.

```
10 PRINT "WELCOME TO LEEDS MIDDLE SCHOOL"
20 GOTO 60
25 PRINT
30 PRINT "HELLO SUPERSTAR"
35 END
40 PRINT "COMPUTERS ARE MY THING"
50 GOTO 100
60 IF A = 5 THEN 140
70 PRINT "COMPUTER WORKSHOP"
80 GOTO 40
90 REM A TRICKY PROGRAM
100 LET A = 5
110 GOTO 60
120 PRINT "AND I'M A SUPERSTAR!"
130 GOTO 25
140 PRINT "TI-99/4A MICROCOMPUTER"
150 PRINT "I CAN DO IT TOO"
160 PRINT "I SPEAK BASIC"
170 GOTO 120
```

ASSIGNMENT 6-1

- Write a program of your choice using conditional (IF-THEN) and unconditional (GOTO) statements.
- Write a counting program, counting to 100 by 10's.

6.6

What We Have Learned

- Relational operators: =, >, <, <>, <=, >=
- IF-THEN
- GOTO (no space between GO and TO)
- Conditional branching
  — If condition is met (i.e., True), branch to designated line in program.
  — If condition is not met (i.e., False), go to next line number in program.
- Unconditional branching
  — GOTO line XX (no conditions or tests required).
  — A GOTO statement, as the name implies, forces the computer to go to a specific statement anywhere in the program.

PRACTICE 9
Using IF-THEN

Part I

1. Enter and RUN the following program:

```
10 LET A = 10
20 IF A = 10 THEN 50
30 PRINT "A DOES NOT EQUAL 10"
40 END
50 PRINT "A EQUALS 10"
```

2. Change Line 10 to LET A = 5 and then RUN it.
3. Change Line 10 to LET A = 3 and then RUN it.

Part II

1. Write a program that assigns a value to variables A and B and prints either "A IS GREATER THAN B" or "B IS GREATER THAN A."
2. Change the values of A and B and RUN the program several times.

PRACTICE 10
Counting Program Using IF-THEN

1. Enter and RUN this program:

```
10 LET J = 0
20 LET J = J + 1
30 PRINT J
40 IF J < 10 THEN 20
```

2. Write a program to count from 1 to 15.
3. Write a program to count to 50 by 5's.
4. Write a program to count to 100 by 10's.
5. Write a program to count from 15 to 30 and PRINT the answers in one column (vertically).

   Example: 15
   16
   17
   18
   and so forth

6. Write a program to count from 20 to 40. PRINT answers horizontally in two columns.

   Example:

20	21
22	23

   and so forth

PART 7

The INPUT Statement

What You Will Learn

1. To explain the purpose and use of the key word **INPUT**.
2. To explain the purpose and use of INPUT with built-in PRINT.
3. To explain the purpose and use of a trailing semicolon in a PRINT statement.
4. To identify and use string variables.
5. To explain the difference between numeric and string variables.
6. To write, enter, and RUN programs that use the concepts of this lesson.

7.1

Description of the INPUT Statement

Statement	Function
1Ø INPUT A	• Causes the computer to stop, PRINT a ?, and wait for you to type in a number. • After you type in a value for A, the computer continues the program when you press the [ENTER] key.

7.2

Example of the INPUT Statement

YOUR ACTION	DISPLAY
1. Type NEW and press ENTER.	TI BASIC READY
2. Type and enter Lines 5 and 10 as shown.	>5 PRINT "THE NUMBER IS" >10 INPUT A
3. Type RUN.	>RUN THE NUMBER IS ? ■ Ⓐ
4. Enter a number (e.g., type 5 and press ENTER).	THE NUMBER IS ? 5 ** DONE ** >■
5. RUN this program several times to get the feel of it.	

Note:

Ⓐ The question mark on the screen means, "It's your turn and I'm waiting."

7.3

The INPUT Statement with Built-In PRINT

YOUR ACTION

1. Type NEW and press [ENTER].

2. Type in the program as shown.

DISPLAY

```
 TI BASIC READY

>5 PRINT "WHAT IS THE NUMBER"
>10 INPUT A
>20 PRINT "THE NUMBER WAS";A
```

3. RUN the program and enter a 5 when the program stops for the INPUT.

```
>RUN
 WHAT IS THE NUMBER
 ? 
 THE NUMBER WAS 5

 ** DONE **
```

4. Type Line 5 to read:

5. RUN the program again; enter a 7 when the program stops for the INPUT.

```
>5 PRINT "WHAT IS THE NUMBER";

>RUN
 WHAT IS THE NUMBER? 
 THE NUMBER WAS 7

 ** DONE **
```

Ⓐ

6. Change Line 5 to read:

```
>5 INPUT "WHAT IS THE NUMBER"
 :A
```

7. Delete Line 10 by typing 10 and pressing [ENTER].

8. RUN the program and enter a 9 when the program stops for the INPUT.

```
>RUN
 WHAT IS THE NUMBER 
 THE NUMBER WAS 9

 ** DONE **
```

Ⓑ

9. LIST the program and RUN it several times, entering a different number each time.

```
>LIST
 5 INPUT "WHAT IS THE NUMBER"
 :A
 20 PRINT "THE NUMBER WAS";A
>
```

Ⓒ

Note:

Ⓐ A semicolon in Line 5 puts the question mark on the same line as the message.

Ⓑ An INPUT statement has a built-in PRINT feature, allowing you to combine a PRINT with an INPUT for the message you want! The resulting RUN was exactly the same except the "?" was not generated.

Ⓒ INPUT with built-in PRINT has the following format:

```
5 INPUT "WHAT IS THE NUMBER":A
```

Line Number — Key Word — Message — Colon — Variable

If you want a question mark (?) at the end of the message, you would make sure it was the last character before you closed the quotes.

7.4

Area of Rectangle Program (Using INPUT Statements)

```
10 REM AREA OF A RECTANGLE
20 REM A = L * W
30 PRINT "THE LENGTH IS"
40 INPUT L
50 PRINT "THE WIDTH IS"
60 INPUT W
70 A = L* W
80 PRINT "THE AREA IS"
90 PRINT A
```

7.5

Area of Rectangle Problem Revisited (Using INPUT Statements)

YOUR ACTION	DISPLAY
1. Type in program Lines 1Ø through 6Ø as shown.	>1Ø REM AREA OF A RECTANGLE PROBLEM >2Ø INPUT "THE LENGTH IS": L >3Ø INPUT "THE WIDTH IS": W >4Ø A = L * W >5Ø PRINT "THE AREA IS"; >6Ø PRINT A Ⓐ
2. Type RUN and press ENTER. 3. Type in the length (say 1Ø) and press ENTER. 4. Type in the width and press ENTER. 5. What is your answer?	THE LENGTH IS 1Ø THE WIDTH IS ■ Ⓑ

Note:

Ⓐ Note the trailing semicolon. It is used to hook Lines 5Ø and 6Ø together.

Ⓑ Note that the program waits for an input from the keyboard. If you don't enter a number or press ENTER, it will just stay at that line until the machine is turned off or reset. Also, note that there is no space between the word "IS" and your entry (1Ø). If you want a space, edit Lines 1Ø and 2Ø so that you can insert a space between the word "IS" and the last quote (").

ASSIGNMENT 7-1

- Write a program to do the following (using INPUT statements):
  a. INPUT your age
  b. INPUT your zip code
  c. INPUT your weight
  d. INPUT your height in inches
  e. PRINT each of the above with the proper labels. For example:
     MY AGE IS 15
     or
     I AM 15 YEARS OLD

7.6

What We Have Learned

- A trailing semicolon hooks two lines together.
- An INPUT statement causes the computer to stop and wait for an input from the keyboard.
- An INPUT statement can have a built-in message to tell you what to enter.

  For example:

  ```
  1Ø INPUT "YOUR AGE": A
  ```

- If you want a question mark (?) to be displayed with your INPUT statement, you must include it inside the quotes.
- If you want a space between your entry (A) and the last character inside the quotes, you must leave a space.

  For example:

  ```
  1Ø INPUT "YOUR AGE? ":A
  ```

7.7

Numeric versus String Variables

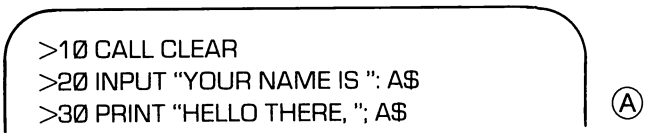
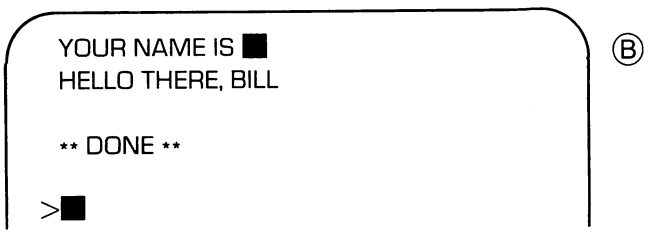
Numeric Variable		Declaration Character*		String Variable
A	+	$	=	A$
A1	+	$	=	A1$
AB	+	$	=	AB$
AZ	+	$	=	AZ$

Note:

⊛ Adding the string declaraton character ($) to the numeric variable allows you to use any numeric variable as a string variable.

7.8

Example of Use of String Variables

YOUR ACTION	DISPLAY
1. Type and enter.	 Ⓐ
2. RUN.	 Ⓑ

Note:

Ⓐ Don't forget to use a colon (:) in Line 20 and a semicolon (;) in Line 30. Also, be sure to leave a space before closing the quotes in Lines 20 and 30.

Ⓑ It will print your name and not "BILL," unless your name is "BILL."

EXERCISE 7-1

YOUR ACTION	DISPLAY
1. Type and enter.	>5 CALL CLEAR >10 INPUT "YOUR FIRST NAME? ": A$ >20 INPUT "YOUR MIDDLE NAME? " :B$ >30 INPUT "YOUR LAST NAME? ":C $ >40 PRINT A$;" ";B$;" ";C$ Ⓐ >50 INPUT "YOUR FULL NAME? ":D $ >60 PRINT D$
2. RUN.	YOUR FIRST NAME? AUBREY YOUR MIDDLE NAME? BRIGHT YOUR LAST NAME? JONES AUBREY BRIGHT JONES YOUR FULL NAME? AUBREY BRIGH T JONES AUBREY BRIGHT JONES ** DONE **

Note:

Ⓐ String variables can be printed together.

○ To insert a space between string variables, you must print a space enclosed in quotes between the variables.

○ A semicolon alone will not cause a space to be printed between string variables.

ASSIGNMENT 7-2

1. RUN and analyze the following program:

```
10 INPUT "YOUR NAME IS": A$
20 INPUT "YOUR HOUSE NUMBER": A
30 INPUT "YOUR STREET NAME": B$
40 INPUT "YOUR ZIP CODE" : B
50 PRINT A$
60 PRINT A ; B$
70 PRINT "ZIP CODE" ; B
```

2. Answer the following questions:
   a. Why were A$ and B$ (string variables) required in Lines 10 and 30?
   b. Why didn't Line 60 contain quotes? (60 PRINT A; " " ; B$)
   c. Why didn't we use the $ symbol (or string declaration character) with A and B in Lines 20 and 40?

SUMMARY

String Variables

- String variables can be assigned to indicate letters, words, and/or combinations of letters, numbers, and special characters.
- It is possible to contain up to 255 characters per string variable.
- String variables can be printed together.
- In a PRINT statement, use " " marks containing a space between string variables to separate them.

PRACTICE 11

Perimeter of Rectangle Problem (Using INPUT Statements)

1. Enter and RUN this program:

```
10 REM PERIMETER OF RECTANGLE PROBLEM
20 INPUT "WHAT IS THE LENGTH":L
30 INPUT "WHAT IS THE WIDTH":W
40 P = 2*L+2*W
50 PRINT "THE PERIMETER IS";P
```

2. Write a new program using INPUT statements to find volume (volume = length × width × height).
3. Include a statement:

   THE VOLUME IS _____.

PRACTICE 12

More INPUT Statement Programs

Part I

1. Write a program using INPUT statements to change meters to centimeters (centimeters = 100 × meters).
2. Include a statement:

   _____ METERS EQUALS _____ CENTIMETERS.

Part II

1. Write a new program using INPUT statements to do the following:
   a. INPUT the year you were born.
   b. INPUT how many brothers you have.
   c. INPUT how many sisters you have.
2. PRINT each with the proper labels. Example:

   I WAS BORN IN _____.
   I HAVE _____ BROTHERS.

PRACTICE 13
String Variables

Part I

1. Enter and RUN the following program:

```
10 INPUT "WHAT IS YOUR NAME? ":A$
20 INPUT "WHAT IS YOUR HOUSE NUMBER? ":A
30 INPUT "WHAT IS YOUR STREET NAME? ":B$
40 INPUT "WHAT IS YOUR ZIP CODE? ":B
50 PRINT A$
60 PRINT A;B$
70 PRINT "ZIP CODE";B
```

2. Answer the following questions:
   a. Why are A$ and B$ (string variables) required in Lines 10 and 30?
   b. Why didn't we use the $ symbol (or string declaration character) with A and B in Lines 20 and 40?

Part II

1. Write a new program using INPUT statements, string variables, and a space between each line. PRINT all information (example: MY BEST FRIEND IS _____) to give the following information:
   a. Your best friend.
   b. Your favorite subject.
   c. Your favorite food.
   d. Your favorite movie star.
   e. Your favorite color.
   f. Your zodiac sign.

PART 8

Using the Calculator Mode

What You Will Learn

1. To review the mathematical operators.
2. To review the order of operations using the M.D.A.S. rule.
3. How to use the TI-99/4A in the calculator mode using variables.

REVIEW

Math Operators

= (Equals)	* (Multiply)
+ (Add)	/ (Divide)
— (Subtract)	∧ (Exponentiation)

Note:

○ Exponentiation (∧) means raising a number to a power like 2^2, 2^3, or 2^4.

REVIEW

Order of Arithmetic Operations

- Multiply → Divide → Add → Subtract (left to right)
  — "My Dear Aunt Sally"
- If parentheses are used:
  — the innermost set of parentheses is simplified first, followed by each successive set outward.
  — the M.D.A.S. order is followed inside all sets of parentheses.

REVIEW

Order of Operations Example (without Parentheses)

- If there are no parentheses, the computer performs operations by going from left to right doing exponentiation operations (∧) first. Then (*) and (/) are done in order from left to right and finally (+) and (—) are done in order from left to right. (Remember M.D.A.S.!)
- Example:

```
4 + 5 * 4 ∧ 3 — 4/2 =
4 + 5 * [64]  — 4/2 =
4 +  [320]    — 4/2 =
4 +   320     — [2] =
[324]         —  2  = [322]
```

REVIEW

Order of Operations Example (with Parentheses)

- If there are parentheses, the computer starts at the inner pair of parentheses and converts everything to a single number. Then the computer repeats the process with the next pair of parentheses working "inside" out.
- Example:

  ((6 + 4) * 2) /4 =
  ([10] * 2) /4 =
  [20] /4 = [5]

Note:

○ The same answer can be worked out first on paper and then on the computer. If you type in

```
PRINT ( (6+4) * 2) /4
```

and press [ENTER], the computer will print out 5.

8.1

Calculator Mode ("Immediate Mode")

- Use PRINT followed by a mathematical expression to have the TI-99/4A act like a calculator.
- You do not use a line number to operate the calculator mode.
- You can use variables in the calculator mode.
- You should type NEW before using variables in the calculator mode. NEW clears all the values variables may have.

ASSIGNMENT 8-1

- Type in each of the following and record the computer's response after you press [ENTER]:

YOU TYPE	RESPONSE
PRINT 4+5	_____
PRINT 6*8–3	_____
PRINT 4+5*4	_____
PRINT 7/2*3–1	_____
PRINT 2^4–5	_____
PRINT (24+3)*(40–20)	_____
PRINT (30–5)–7+(3*8)	_____
PRINT 4^3–(34–30)^2	_____

ASSIGNMENT 8-2

- Type in each of the following and record the computer's response after you press [ENTER] :

YOU TYPE	RESPONSE
NEW	_____
PRINT A	_____
PRINT B	_____
A=5	_____
B=12	_____
C=4	_____
PRINT A+B–C	_____
PRINT C*A–B	_____
PRINT C^A–B	_____
PRINT A*B–B*A	_____
NEW	_____
PRINT A	_____
PRINT B	_____
PRINT C	_____

8.2
What We Have Learned

- The command NEW clears the values of all variables and erases any program. Numeric variables will then have a value of Ø.
- To use the calculator mode, type PRINT followed by a mathematical expression and press [ENTER]. The answer to the mathematical expression will then be printed.
- Variables can be assigned values in the calculator mode.
- You can evaluate expressions using variables in the calculator mode.
- The TI-99/4A evaluates expressions in the calculator mode using the M.D.A.S. rule.

PRACTICE 14
Calculator Mode

Part I

1. Use the TI-99/4A in calculator mode to solve the following:
   a. 25*8/2
   b. (25+6)−7+(2*4)
   c. 7/2*5*2^4

Part II

1. Assign the following values in the calculator mode (remember to type NEW):
   a. A=6
   b. B=2
   c. C=1Ø
   d. D=3
2. Use the calculator mode to solve the following with the values from 1:
   a. A*B−C+D
   b. D^B*A−C
   c. (A+D)*C^B

PART

Using the Cassette Recorder

What You Will Learn

1. How to use the cassette as an output device to save information stored in memory.
2. How to use the cassette as an input device to load information from tape to memory.
3. To explain and use the commands **SAVE, OLD** (Load).
4. To make critical settings on the tape recorder and to practice using the recorder.

9.1

The Cassette Recorder as an I/O Device

- The cassette tape recorder is an input/output (I/O) device that allows you to "save" information on cassette or "load" information from cassette.
  - — When you have typed a long program and wish to save it, you can save (SAVE) it on cassette.
  - — When you are ready to use it again, you can load (OLD) it from the cassette.
  - — After you have saved your program, you should check it for recording errors. You can do this with a Check Tape option.

Note:

○ You can save only your program on cassette (not the program output).
○ Refer to the *TI User's Reference Guide* for tips on using the recorder.

9.2

SAVE Command (with Check Tape Option)

- Writes (outputs) a copy of the current program from memory to the tape cassette recorder.
- Format: SAVE <space> file name
  — Where "file name" will be either a "CS1" or CS2" (without quotes).
  — Where "CS1" selects cassette #1 and "CS2" selects cassette #2.
- Example:

Command	Meaning
SAVE CS1	Write (save) current program on Cassette #1.
SAVE CS2	Write (save) current program on Cassette #2.

Screen Prompt	Meaning
CHECK TAPE (Y OR N)	When this message (prompt) occurs during a SAVE processing, the computer is asking you if you wish to verify that the data saved is correct. Your response should be: Y (yes). (You do have the option of typing N, however.) Y or N *must* be in upper case.

- If you wish to verify that the recording made by the SAVE command is accurate, you can let the computer check your tape to make sure your tape is recorded properly. *It is highly recommended that you verify all data.*
  — To verify, the computer reads and compares the program and tape with the program in memory.

9.3

Saving a Program from Memory on the Cassette Recorder (Using the Recorder as an Output Device)

YOUR ACTION	DISPLAY
1. Type and [ENTER] the program shown.	>10 PRINT "THIS IS A TEST PRO GRAM" >20 PRINT "THIS PROGRAM WILL BE SAVED ONTO TAPE."
2. Place a blank tape in recorder. 3. Rewind tape to the beginning. 4. Type SAVE command as shown.	>SAVE CS1
5. Press [ENTER] .	* REWIND CASSETTE TAPE CS1 THEN PRESS ENTER
6. If tape is rewound, then press [ENTER] .	* PRESS CASSETTE RECORD CS1 THEN PRESS ENTER
7. Press RECORD and then press [ENTER] .	* RECORDING
	* PRESS CASSETTE STOP CS1 THEN PRESS ENTER
8. Type Y (upper-case "Y") for Yes .	* CHECK TAPE (Y OR N)? Y
9. Press [ENTER] .	* REWIND CASSETTE TAPE CS1 THEN PRESS ENTER
10. Press [ENTER] .	* PRESS CASSETTE PLAY CS1 THEN PRESS ENTER
	* CHECKING
	* DATA OK
	* PRESS CASSETTE STOP CS1 THEN PRESS ENTER
11. Press STOP and [ENTER] .	>■

9.4
OLD (Load) Command

- Loads (inputs) a previously saved program from tape to memory.
- Format: OLD file name
  — Where "file name" must be CS1, since you can use only CS1 to load data. However, either CS1 or CS2 can be used to SAVE data.
- Example:

Command	Meaning
OLD <space> CS1	Load program previously saved on cassette #1 into memory.

9.5

Loading a Program from Tape Cassette to Memory (Using the Tape Cassette as an Input Device)

YOUR ACTION	DISPLAY
1. Make certain that the tape is rewound to the beginning, and also that you type NEW.	>NEW TI BASIC READY
2. Type the command shown.	>OLD CS1
3. Press the [ENTER] key.	* REWIND CASSETTE TAPE CS1 THEN PRESS ENTER
4. Press PLAY on the cassette, then press [ENTER] .	* PRESS CASSETTE PLAY CS1 THEN PRESS ENTER * READING * DATA OK
5. Press STOP on the cassette, then press [ENTER] .	* PRESS CASSETTE STOP CS1 THEN PRESS ENTER >■

Note:

○ After you type the OLD command and press [ENTER] , the computer will begin printing instructions on the screen to help you through the procedure.

9.6

Loading a Program from Tape Recorder to Memory (Error Procedure)

- If the computer did not successfully read your program into memory, an error will occur and you may choose either of these options:

YOUR ACTION	DISPLAY
1. Press upper-case R to repeat the reading procedure.	* ERROR - NO DATA FOUND PRESS R TO READ PRESS E TO EXIT
2. Press upper-case E to exit from the reading procedure.	* I/O ERROR 56

Note:

○ Before repeating, check the cassette to make certain it is connected in accordance with the directions given in the *User's Reference Guide*. Pages I-9 through I-12 and page 42 cover this topic.

PRACTICE 15

Using the Computer to Solve Problems

1. Write a program to solve the following problem. Include a PRINT statement in your program to describe your answer (output).
   The total enrollment at Armstrong High School is 1,264. There are 367 freshmen, 322 sophomores, and 298 juniors. How many seniors are there?
2. Write a new program using INPUT statements to solve the same problem.
   (That is, you should use an INPUT statement for the total enrollment, number of freshmen, number of sophomores, number of juniors.)

PRACTICE 16

Finding the Average Problems

1. Write a program to solve the following problem. Include a PRINT statement in your program to describe your answer.
   The weights of three boys are 14Ø lb., 15Ø lb., and 13Ø lb. What is their average weight?
2. Write a new program using INPUT statements to solve the same problem.
   (That is, you should use an INPUT statement for the weight of each of the three boys.)

PRACTICE 17

Using the Computer to Solve Problems

1. Write two programs to solve the following problems. Label your answers.
2. Over a period of six years Mr. Smith drove his car 53,862 miles. What was the average distance each year?
3. After 12 dozen bulbs were sold, how many of the 1ØØØ bulbs were left?

PART

Using FOR-NEXT...STEP Statements

10

What You Will Learn

1. To explain the purpose and use of key words **FOR-NEXT...STEP.**
2. To explain the purpose and use of the terms increment, decrement, initialize.
3. To compare key words GOTO, IF-THEN, FOR-NEXT and explain how they relate to one another.
4. To explain the purpose and use of timer loops.

10.1
FOR-NEXT Statement

- Allows the computer to do the same thing over and over a number of times (and do it very fast!)

YOUR ACTION	DISPLAY
1. Type and enter program as shown.	>5 CALL CLEAR >10 FOR J = 1 TO 10 >20 PRINT " AUBREY" ; J >30 NEXT J
2. Type RUN and press ENTER.	AUBREY 1 AUBREY 2 AUBREY 3 AUBREY 4 AUBREY 5 AUBREY 6 AUBREY 7 AUBREY 8 AUBREY 9 AUBREY 10 ** DONE ** >■

10.2
FOR-NEXT...STEP Loop

YOUR ACTION	DISPLAY
1. Retype and enter Line 1Ø of resident program as shown. Ⓐ	>1Ø FOR J = 1 TO 1Ø STEP 3 Ⓑ
2. Type RUN and press ENTER.	AUBREY 1 AUBREY 4 AUBREY 7 AUBREY 1Ø ** DONE ** >■

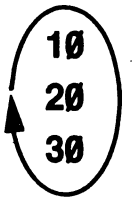
Note:

Ⓐ Resident means program currently in memory.

Ⓑ If "STEP" is not included in the statement, an increment of 1 is assigned by the computer (i.e., STEP 1).

10.3

Example of Program Statements Using FOR-NEXT...STEP



```
10 FOR J = 10 TO 1 STEP -1
20 PRINT J ;
30 NEXT J
```

```
RUN
DISPLAY READS:
10 9 8 7 6 5 4 3 2 1
```

10.4

Analysis of FOR-NEXT...STEP Statements

LINE NO.	KEY WORD	COUNTER VARIABLE		INITIAL VALUE		FINAL VALUE	INCREMENT/ DECREMENT
1Ø	FOR	J	=	1Ø	TO	1	STEP –1
2Ø	PRINT	J ;					
3Ø	NEXT	J					

- The FOR-NEXT...STEP loop works as follows: The first time the FOR statement is executed, the counter is set for the initial value "1Ø." Then it executes Line 2Ø (PRINT J). When the program reaches Line 3Ø (NEXT J), the counter is decremented by the amount specified (STEP –1). If this step has a positive value, the counter is incremented by the amount specified (e.g., STEP 2 means increment by 2's).

10.5

Comparison of Program Loops (Listings)

A. GOTO (Unconditional Loop)

```
5 CALL CLEAR
10 LET J = 0
20 LET J = J + 1
30 PRINT "AUBREY" ; J
40 GOTO 20
```

- This program loops forever (or until you stop it).

B. IF-THEN (Conditional Loop)

```
5 CALL CLEAR
10 LET J = 0
20 J = J + 1
30 IF J>6 THEN 99
40 PRINT "AUBREY" ; J
50 GOTO 20
99 END
```

- This program loops 6 times.

C. FOR-NEXT (Conditional Loop)

```
5 CALL CLEAR
10 FOR J = 1 TO 6
20 PRINT "AUBREY" ; J
30 NEXT J
99 END
```

- This program loops 6 times.

10.6

Comparison of Program Loops (Outputs)

A. "DUMB LOOP"*	B. "SMART LOOP"	C. "SMART LOOP"
AUBREY 1	AUBREY 1	AUBREY 1
AUBREY 2	AUBREY 2	AUBREY 2
AUBREY 3	AUBREY 3	AUBREY 3
AUBREY 4	AUBREY 4	AUBREY 4
AUBREY 5	AUBREY 5	AUBREY 5
AUBREY 6	AUBREY 6	AUBREY 6
AUBREY 7		
AUBREY 8		
AUBREY 9		
AUBREY 10		
AUBREY 11		
AUBREY 12		
AUBREY 13		
AUBREY 14		
AUBREY 15		
AUBREY 16		

Note:

* Press the FCTN CLEAR keys to get out of this loop.

SUMMARY

FOR-NEXT...STEP

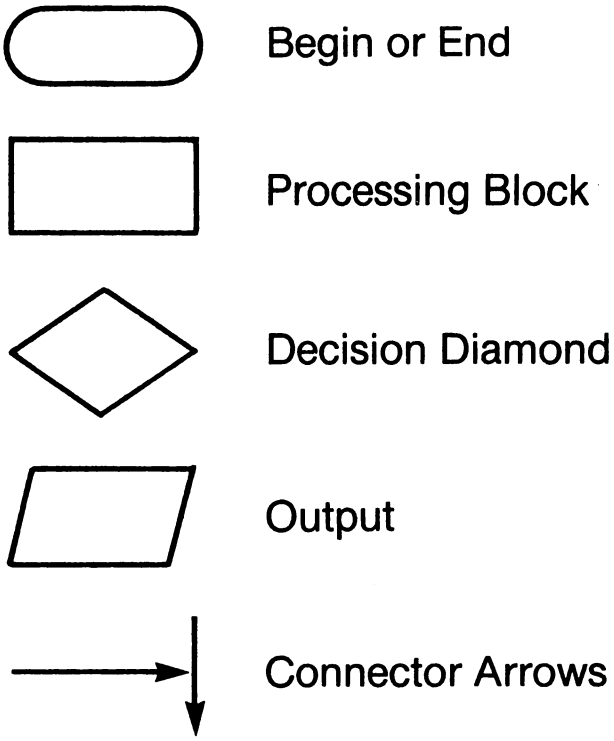
- FOR-NEXT is always used as a pair.
- If the key word "STEP" is not used, the increment of 1 is assumed.
- If the STEP has a negative value, the counter is decremented. For example:

```
20 FOR J = 10 TO 1 STEP -1
```

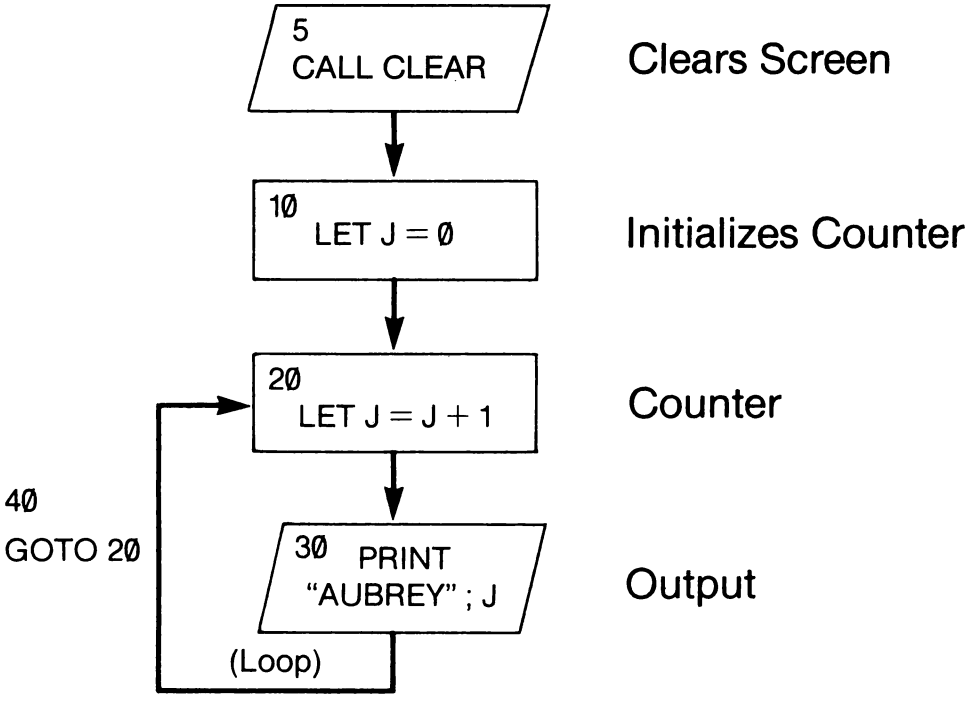
- If the STEP has a positive value, the counter is incremented. For example:

```
20 FOR J = 4 TO 10 STEP 2
```

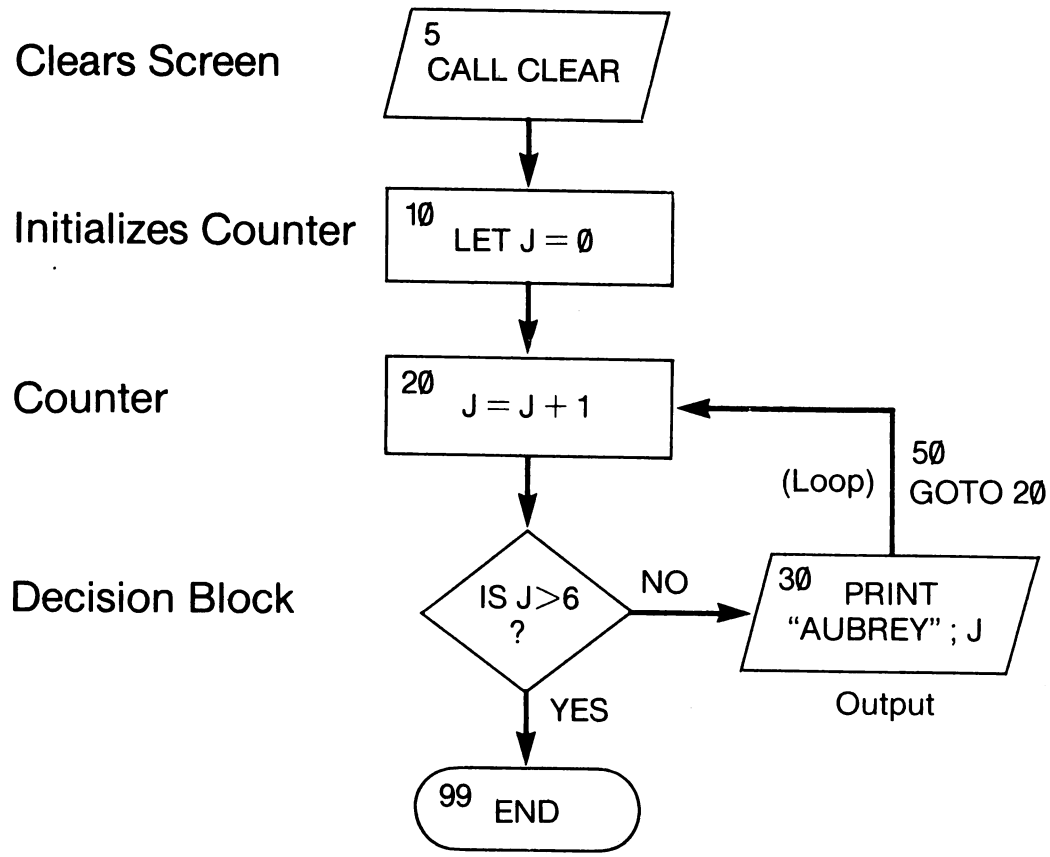
10.7
Flowchart Symbols



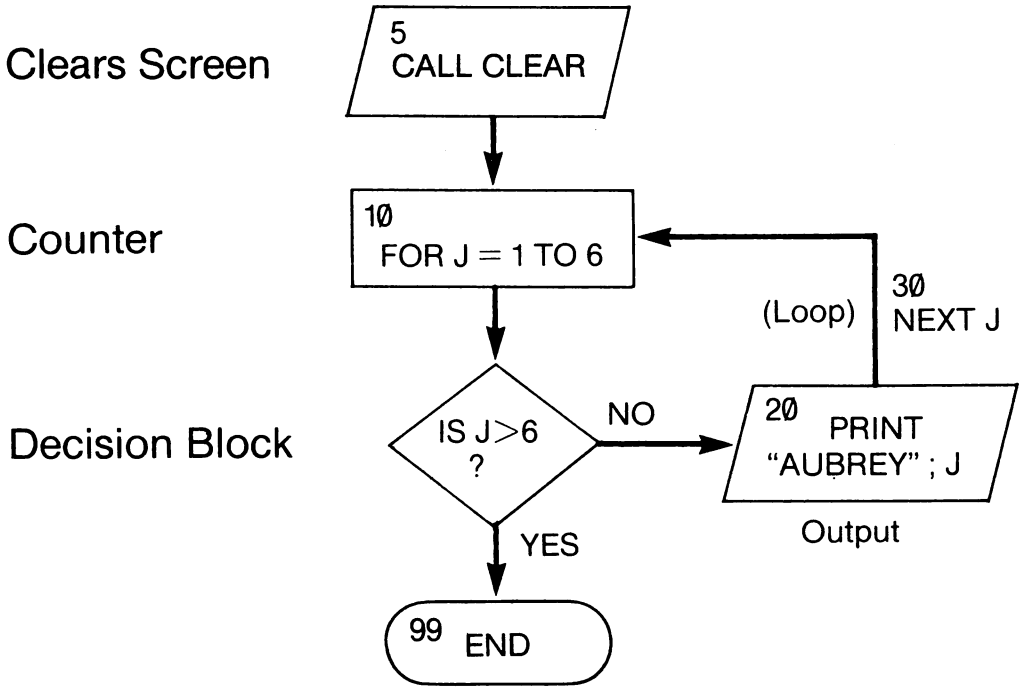
10.8
GOTO Loop



10.9
IF-THEN Loop



10.10
FOR-NEXT Loop



Note:

○ FOR-NEXT work together as a counter.

10.11
Timer Loop

- The TI-99/4A can do approximately 330 FOR-NEXT loops per second.
- Example:

```
5 REM 10 SECOND TIMER PROGRAM
10 PRINT "TIMER PROGRAM COUNTING"
20 FOR X = 1 TO 3300
30 NEXT X
40 PRINT "TIMER PROGRAM ENDED"
```

- You don't believe the TI-99/4A can count? Well, try it! Type in the above program and RUN. Don't forget to use your watch!

ASSIGNMENT 10-1

1. Type in and RUN the following program:

```
5 CALL CLEAR
10 PRINT "INPUT A VALUE FOR N"
12 PRINT
15 INPUT "ENTER 1500, 2500, 3500 OR 7500":N
20 CALL CLEAR
25 PRINT "THIS IS A DEMONSTRATION OF"
30 PRINT
35 FOR J=1 TO N
37 NEXT N
40 PRINT "USING A FOR-NEXT TIMER LOOP"
45 PRINT
47 PRINT
50 FOR J=1 TO N
55 NEXT J
60 PRINT "IF YOU WISH TO CHANGE THE DISPLAY'S SPEED"
65 PRINT
67 PRINT
70 FOR J=1 TO N
75 NEXT J
80 PRINT "CHANGE THE VALUES OF N IN THE FOR-NEXT LOOP"
85 PRINT
87 PRINT
90 FOR J=1 TO N
95 NEXT J
100 PRINT "IF YOU WISH TO STOP THIS DISPLAY"
105 PRINT
107 PRINT
110 FOR J=1 TO N
115 NEXT J
120 PRINT "PRESS THE 'FCTN' and 'CLEAR' KEYS"
130 FOR J=1 TO N
135 NEXT J
140 GOTO 20
```

2. Make certain that you understand this program and can explain it to your teacher.

PRACTICE 18

Counting Programs Using IF-THEN and FOR-NEXT

1. Using IF-THEN, write a program to count by 5's from 5Ø to 5.
   a. Written vertically
   b. Written horizontally
2. *Do not* type NEW (that is, SAVE the program above).
3. Using FOR-NEXT, write a program to count by 5's from 5Ø to 5 written horizontally.
   Note: Start your second program at Line 1ØØ. That is, type Line 1ØØ as follows: 1ØØ PRINT.
   (Of course, this is to insert one space between your outputs.)
4. How many program lines (excluding Line 1ØØ) did it take using FOR-NEXT? _____
   How many using IF-THEN? _____
5. What can you conclude from this task?

PRACTICE 19

Using IF-THEN and FOR-NEXT Statements

1. Using IF-THEN, write a program to generate all the even numbers between 11 and 51 from the smallest to the largest (that is, 12, 14, 16, and so forth).
2. *Do not* type NEW.
3. Using FOR-NEXT, write a program that generates the same numbers and PRINT them horizontally. (**Note:** Start at Line 1ØØ. Type Line 1ØØ as → 1ØØ PRINT.)
4. Type NEW and enter.
5. Using IF-THEN, write a program to generate all even numbers between 11 and 51 from the largest to the smallest.
6. Do the same using FOR-NEXT.

PART 11

Reading Data

What You Will Learn

1. To explain the purpose and use of the key words **READ, DATA, RESTORE.**
2. To compare the three different ways you have learned to enter data into the TI-99/4A.
3. To write, enter, and RUN programs using READ-DATA and RESTORE.

11.1

READ-DATA Statements

READ-DATA statements are much more efficient than INPUT or LET statements when you have lots of data to enter.

- Explain that the READ-DATA statement pair is just another way to enter data into the computer.

11.2

Ways of Entering Data into the Computer

Built-In	From Keyboard	READ-DATA Combination
10 LET A = 5	10 INPUT A	10 DATA 5 20 READ A
• Builds value into the program	• Allows you to enter data through the keyboard	• DATA statement contains the value • READ statement assigns the value to the variable named

Note:

◯ Data lines can be read only by READ statements.

◯ READ-DATA work together to enter data into the computer.

11.3
READ-DATA Example

```
                    5 REM READ—DATA EXAMPLE
DATA Statement      1Ø DATA     (1) , (2) , (3) , (4) , (5)
                                 ↓     ↓     ↓     ↓     ↓
READ Statement      2Ø READ      A  ,  B  ,  C  ,  D  ,  E
PRINT Statement     3Ø PRINT     A  ,  B  ,  C  ,  D  ,  E
```

Note:

- ◯ Each variable in a READ statement must have a corresponding value in a DATA statement.
- ◯ Each READ statement can read a number of pieces of data if each variable is separated by a comma.
- ◯ Data lines can be used only by READ statements.

EXERCISE 11-1
READ-DATA

YOUR ACTION	DISPLAY
1. Type and enter.	10 DATA 1,2,3,4,5 20 READ A,B,C,D,E 30 PRINT A,B,C,D,E
2. Type RUN and press [ENTER].	1 2 3 4 5

Note:

- ◯ The display shows that all five pieces of data in Line 10 were read by Line 20, assigned letters A through E, and printed by Line 30.
- ◯ Data lines are always read left to right by READ statements.

SUMMARY

DATA Statement

- Key word that lets you store data inside your program to be accessed (read) by READ statements.
  - — Data items will be read sequentially starting with the first item in the first DATA statement and ending with the last item in the last DATA statement.
  - — Items in data list may be string or numeric values.
  - — If string values include leading blanks, colons, or commas, you must enclose these values in quotes.
  - — DATA statements must match up with the variable types in the corresponding READ statement.
  - — DATA statements may appear anywhere it is convenient in a program.
- Example:

```
10 DATA "JONES, A.B.","SMITH, R.J."
20 DATA LEEDS MIDDLE SCHOOL, COMPUTERS
30 DATA 125, 250, 750, 1000
```

Note:

○ Quotes are used in Line 10 because data contain commas.

SUMMARY

READ Statement

- Key word that instructs the computer to read a value from a DATA statement and assign that value to the specified variable.
  - — The first time a READ statement is executed, the first value in the first DATA statement is used; the second time, the second value in the DATA statement is used. When all the items in the first DATA statement are used (read), the next READ will use the first value in the second DATA statement, and so on.
  - — A DATA ERROR occurs if there are more attempts to READ than there are data items.
- Example:

```
10 DATA "JONES, A.B.","SMITH, R.J."
20 DATA LEEDS MIDDLE SCHOOL,COMPUTERS
30 DATA 125,250,750,1000
40 READ A$,B$,C$,D$,A,B,C,D
```

Note:

○ There are eight variables in the READ statement and eight items in the DATA statements.

ASSIGNMENT 11-1

1. Type and enter the following program:

```
5 CALL CLEAR
10 PRINT "NAME","GRADE"
20 READ A$
30 IF A$="END" THEN 99
40 READ G
50 IF G > 75 THEN 20
55 PRINT A$,G
60 GOTO 20
70 DATA "GRAY,BILL",95,"JONES,A.B.",65
80 DATA "JONES,A.C.",100,"SMITH,R.L.",70
90 DATA "EPPS,S.W.",60,"WELLS,DAVE",100,END
99 PRINT "END OF LIST"
100 END
```

2. Predict the output of the program.
3. Why were quotes used in the DATA statements?
4. RUN the program and record the results.

11.4
RESTORE

- Key word that causes the next READ statement executed to start over with the first DATA statement.
  - — This lets your program reuse the same data lines.
  - — Sometimes it is necessary to READ the same data more than once without having to RUN the complete program again; therefore, RESTORE is used.
  - — Whenever the program comes to RESTORE, all data lines are restored to their original unread condition, both those lines that have been used and those that have not been used. This allows all data to be available for reading again, starting with the first data item in the first data line.

Note:

○ Remember that each piece of data in a data line can be read only once each time the program is RUN. The next time a READ statement requests a piece of data, it will READ the next piece of data in the data line, or, if data on that line are all used up, it will go to the next data line and start reading it. Therefore, the RESTORE statement is needed if the same data is to be used more than once in the same program.

11.5

Illustration of the RESTORE Statement

```
10 DATA (1), 2, 3, 4, 5
20 ...       ↓   FOR N = 1 TO 5
30 READ A
40 PRINT A;
45 RESTORE
50 NEXT N
RUN
  1  1  1  1  1
```

Note:

- RESTORE caused data Line 10 to be restored to its original unread condition, making all data available for reading again.
- Since there is only one READ variable, A, it starts with the first piece of data, 1 in this case.

EXERCISE 11-2

RESTORE and READ-DATA Statements in a FOR-NEXT Loop

YOUR ACTION	DISPLAY
1. Type and enter.	>10 DATA 1,2,3,4,5 >20 FOR N = 1 TO 5 >30 READ A >40 PRINT A ; >50 NEXT N
2. Type RUN and press [ENTER].	1 2 3 4 5 ** DONE **
3. Insert Line 45 (type and enter).	>45 RESTORE Ⓐ
4. LIST the program.	>LIST 10 DATA 1,2,3,4,5 20 FOR N = 1 TO 5 30 READ A 40 PRINT A ; 45 RESTORE 50 NEXT N
5. Type RUN and press [ENTER].	>RUN 1 1 1 1 1 ** DONE ** Ⓑ

Note:

Ⓐ This restores data line to its original unread condition.

Ⓑ Computer reads first data item over and over.

SUMMARY

READ-DATA, RESTORE

- READ-DATA
  - — These key words are used to enter lots of data into the computer.
  - — READ-DATA statements are common in programs.
- READ
  - — Each variable in a READ statement must have a corresponding value in a DATA statement or a DATA ERROR will occur.
- DATA
  - — DATA statements can be placed anywhere in a program.
  - — DATA statements can be used only by READ statements.
  - — If more than one piece of data is placed in a DATA statement, they must be separated by commas.
  - — DATA statements are read from left to right by READ statements.
- RESTORE
  - — This key word is used to return DATA statements to their original, unread state so they can be used again.

PRACTICE 20
READ-DATA

1. Type and enter the following program:

```
5 CALL CLEAR
10 PRINT "NAME","GRADE"
20 READ A$
30 IF A$="END" THEN 99
40 READ G
50 IF G < 75 THEN 20
55 PRINT A$,G
60 GOTO 20
70 DATA "GRAY,BILL", 95,"JONES, A.B.",65
80 DATA "JONES,A.C.",100,"SMITH,R.L.",70
90 DATA "EPPS, S.W.",60,"WELLS, DAVE",100,END
99 PRINT "END OF LIST"
100 END
```

2. Predict the output of the program.
3. Why were quotes used in the DATA statements?
4. RUN the program and record the results.

PART 12

Video Display Graphics

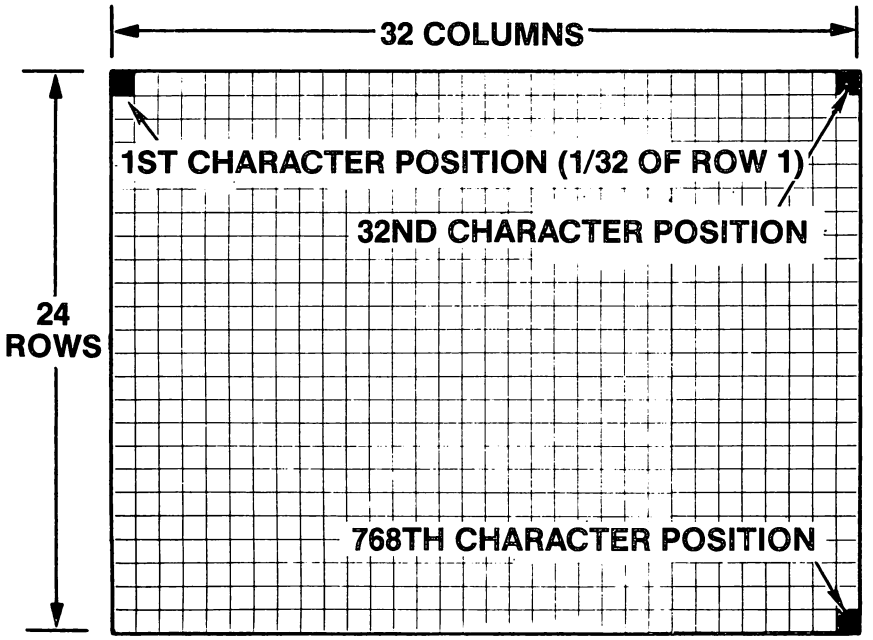
What You Will Learn

1. To explain the purpose and use of key words **CALL HCHAR, CALL VCHAR, CALL CHAR,** and **TAB**.
2. To become familiar with the layout of the TI-99/4A display via the video display worksheet.
3. To draw pictures and letters on the screen.
4. To write, enter, and RUN programs utilizing all the concepts learned in this lesson.

Note:

- ○ The TI-99/4A provides the user with an unlimited number of possibilities for graphic application. You should experiment with graphics. This lesson will introduce you to some of the basic concepts and features used on the TI-99/4A, but we will only "scratch the surface." You will find out by experimenting what other kinds of things can be done with graphics on the TI-99/4A.
- ○ This lesson assumes a black-and-white monitor is used. If you are using a color monitor, refer to the TI-99/4A *User's Reference Guide* for additional information.

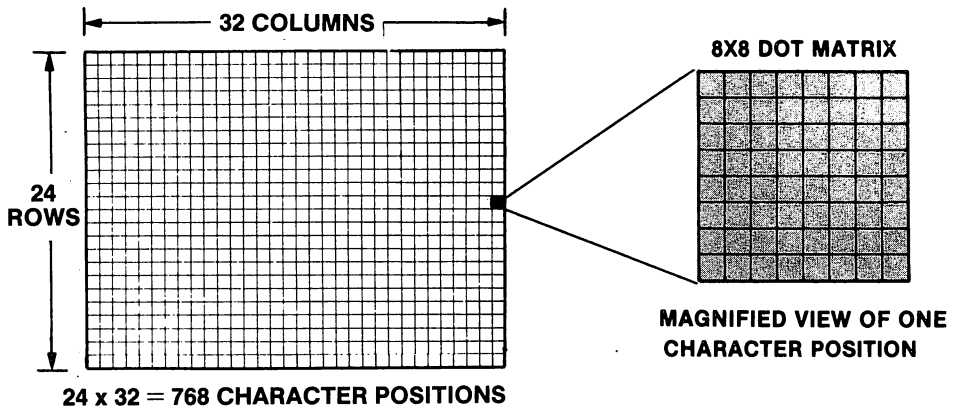
12.1
Video Display Layout



- The display has 768 character positions arranged as 24 rows, with 32 characters per row (24 × 32 = 768 characters).
  - — One-thirty-second of each display line is a character position.
  - — Each character position is an 8x8 dot matrix (dot block) which is used to make characters (see next page).
  - — Some display screens may not show the two leftmost and two rightmost characters; therefore, your graphics may be more satisfactory if you use Columns 3-3Ø and ignore Columns 1 and 2 on the left and 31 and 32 on the right.

12.2

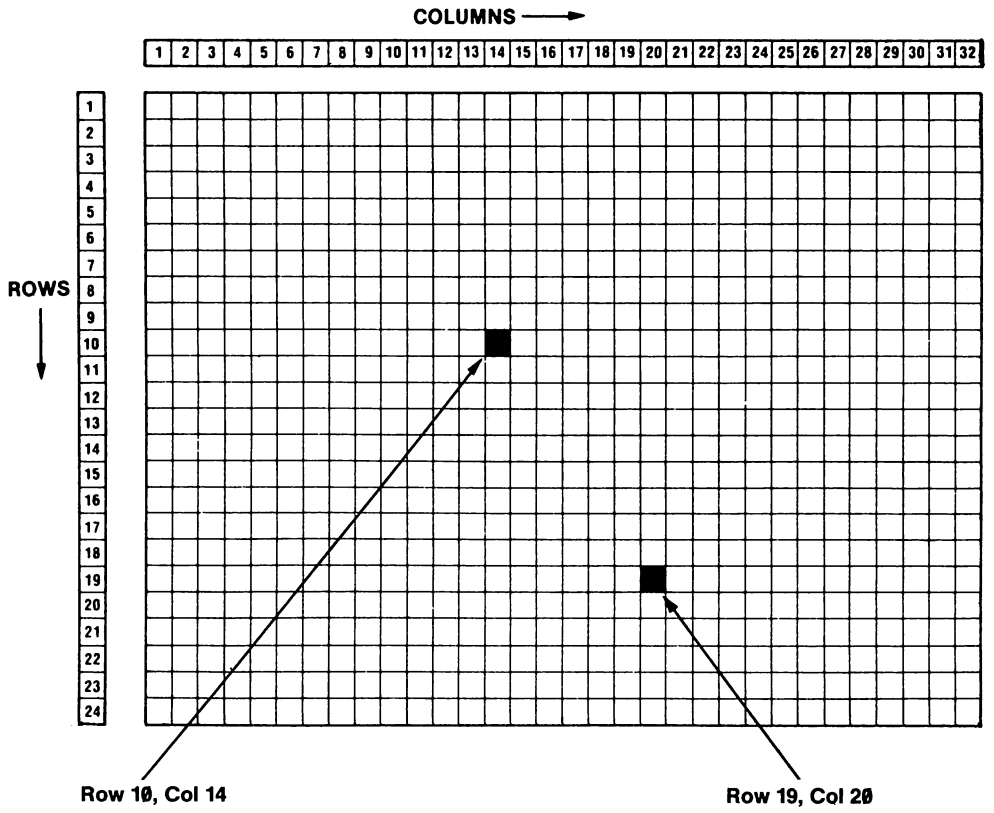
Illustration of Dot Matrix for One Character Position



Note:

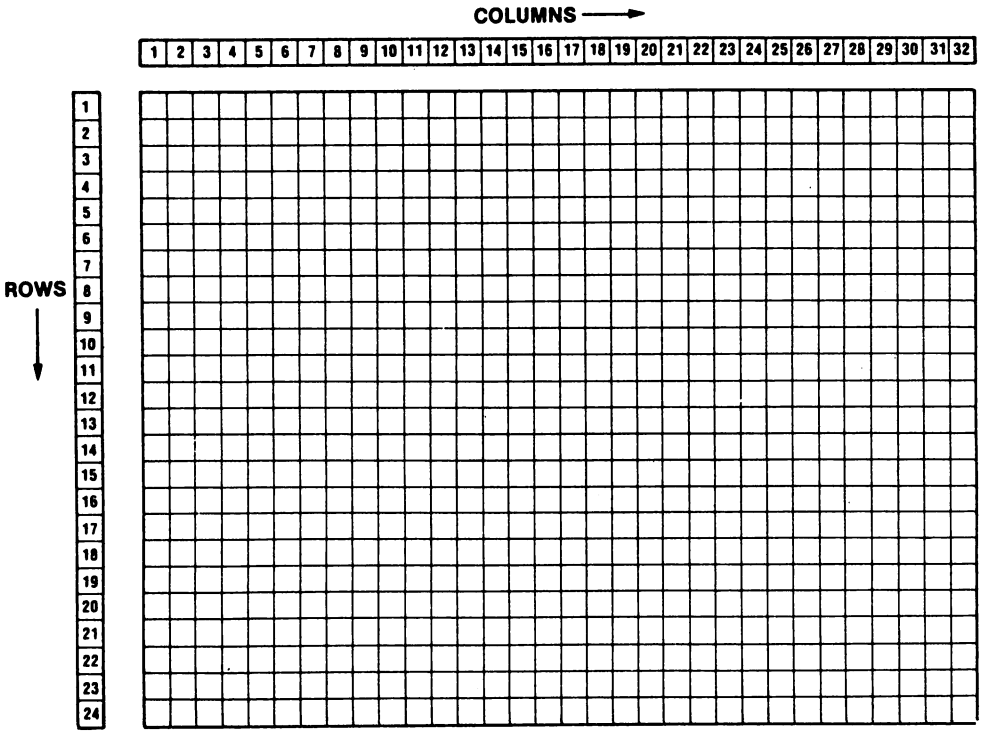
○ There are 64 dots (maximum) available in a single character position. That is, one character position = 8x8 = 64 dots (maximum) are available for generating a single character.

12.3
Description of Video Display Worksheet



- Think of the video display as a grid of square blocks consisting of 32 columns and 24 rows.
  - — Each square on the grid is identified or located by two values called coordinates (Row, Column).
  - — For example, the coordinates 10,14 mean the tenth row and the fourteenth column.

12.4
Video Display Worksheet for Graphics



EXERCISE 12-1

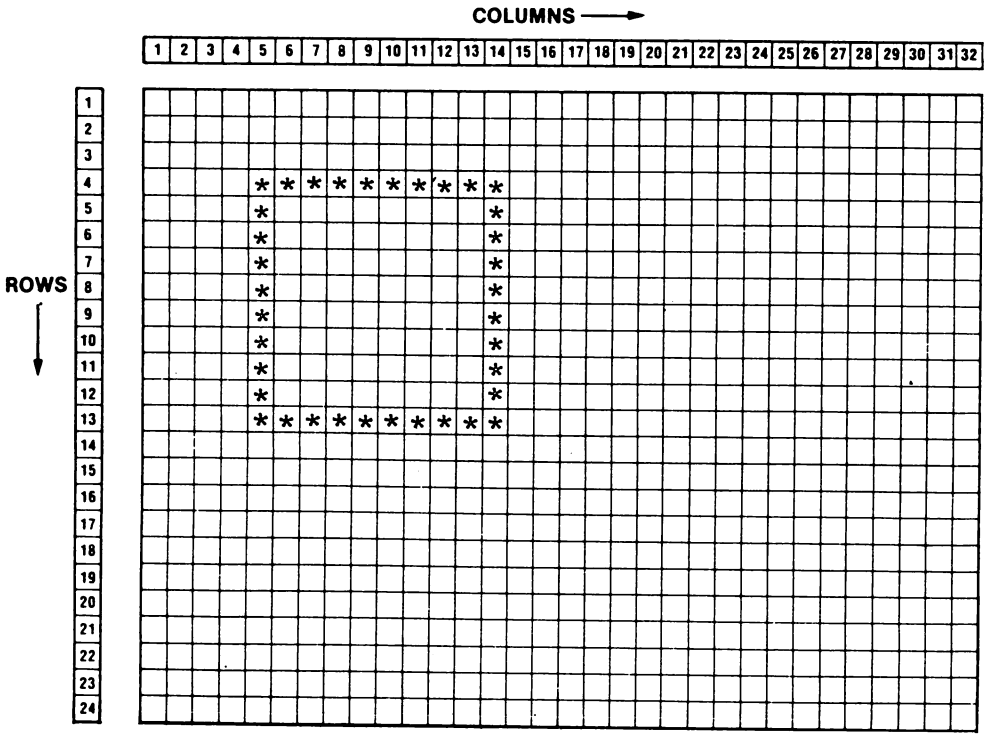
Using the Video Display Worksheet

- Locate the following X(Row), Y(Col) coordinates on your worksheet and place an "*" (asterisk) in each location:

Row,Col	Row,Col	Row,Col	Row,Col
4,5	5,5	13,6	4,14
4,6	6,5	13,7	5,14
4,7	7,5	13,8	6,14
4,8	8,5	13,9	7,14
4,9	9,5	13,1Ø	8,14
4,1Ø	1Ø,5	13,11	9,14
4,11	11,5	13,12	1Ø,14
4,12	12,5	13,13	11,14
4,13	13,5	13,14	12,14

Note:

○ Your results should be a 1Ø × 1Ø square as shown below:



12.5

TI Graphics Statements

- A special set of statements is used to provide color, graphics, sound, and other capabilities not usually found in the BASIC programming language. Whenever you want to use one of these statements, you CALL for it by name and supply a few specifications. The statement then takes over and performs its task. You have already used one of these statements, CALL CLEAR in earlier lessons. The format of statements covered in this lesson are:
  - — CALL HCHAR (Row, Column, Char Code, Repetitions)
  - — CALL VCHAR (Row, Column, Char Code, Repetitions)
  - — CALL CHAR (Char Code, Pattern-Identifier)

12.6

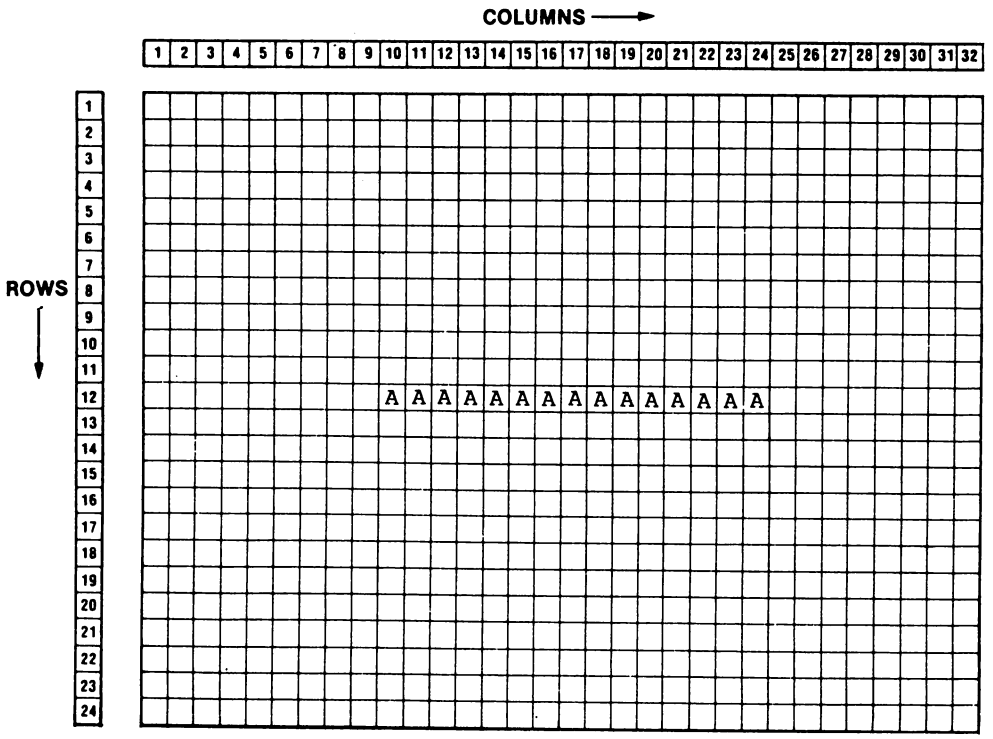
CALL HCHAR (Horizontal Character)

- Displays a character anywhere on the screen and, optionally, repeats it horizontally.
- Format: CALL HCHAR (Row, Column, Char Code, Repetitions)
  - — Where "Row" is the starting row number on the screen. The value for row number can be from 1-24, inclusive.
  - — "Column" is the starting column number on the screen. The value for column number can be from 1-32, inclusive.
  - — "Char Code" or character code is a unique code for any one of the 26 letters of the alphabet, the numbers Ø-9, and certain other symbols, such as the asterisk (*), the plus and minus signs (+ and −), and the slash (/). The range for char code can be specified from Ø-32767, inclusive, but the computer will convert the value specified to a range of Ø-225. (Refer to Section 12.8.)

— "Repetitions" or number of repetitions is optional. It determines the number of times you want a character to be repeated horizontally on the screen. The range for the number of repetitions is from Ø-32767, inclusive.

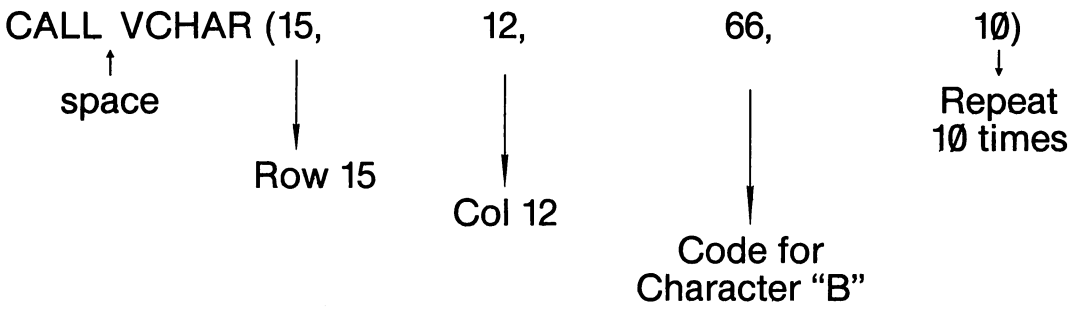
- Example:

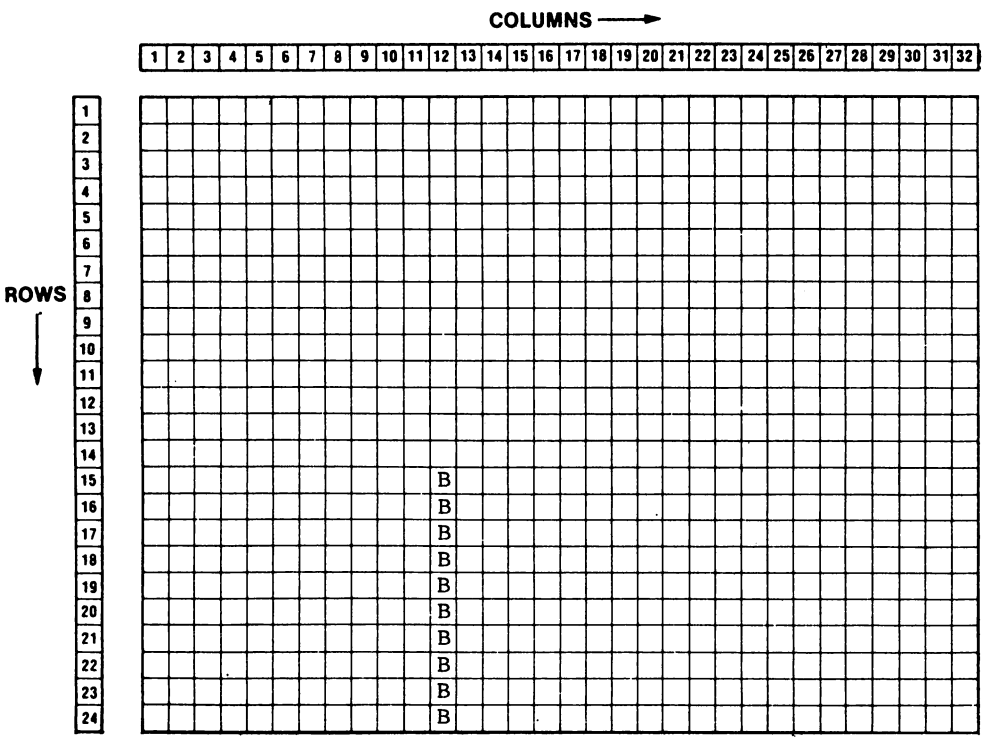
```
CALL HCHAR (12,   1Ø,   65,   15)
     ↑          ↓      ↓      ↓
   space      Row 12  Col 1Ø  Repeat
                              15 times
                      Code for
                    Character "A"
```



12.7
CALL VCHAR (Vertical Character)

- Displays a character anywhere on the screen and, optionally, repeats it vertically. The computer will display the character beginning at the specified row and column. If the character is to be repeated, it will continue down the screen.
- Format: CALL VCHAR (Row, Column, Char Code, Repetitions)
  - — Where "Row," "Column," "Char Code," and "Repetitions" have the same meaning as for CALL HCHAR.
- Example:





12.8

Character Codes

- All characters that print on the screen (letters, numbers, and symbols) are identified by numeric character codes. The standard characters are represented by character codes 32-127. There are 32 additional codes (128-159) available for use in defining special characters for graphics programs. These codes are shown below:

Code	Char	
32		(space)
33	!	(exclamation point)
34	"	(quote)
35	#	(number or pound sign)
36	$	(dollar)
37	%	(percent)
38	&	(ampersand)
39	'	(apostrophe)
40	(	(open parenthesis)
41	)	(close parenthesis)
42	*	(asterisk)
43	+	(plus)
44	,	(comma)
45	−	(minus)
46	.	(period)
47	/	(slant)
48	0	
49	1	
50	2	
51	3	
52	4	
53	5	
54	6	
55	7	
56	8	
57	9	
58	:	(colon)
59	;	(semicolon)
60	<	(less than)
61	=	(equals)
62	>	(greater than)
63	?	(question mark)
64	@	(at sign)
65	A	
66	B	
67	C	
68	D	
69	E	
70	F	
71	G	
72	H	
73	I	
74	J	
75	K	
76	L	
77	M	
78	N	
79	O	
80	P	
81	Q	
82	R	
83	S	
84	T	
85	U	
86	V	
87	W	
88	X	
89	Y	
90	Z	
91	[	(open bracket)
92	\	(reverse slant)
93	]	(close bracket)
94	^	(exponentiation)
95	_	(line)
96	`	(grave)
97	a	
98	b	
99	c	
100	d	
101	e	
102	f	
103	g	
104	h	
105	i	
106	j	
107	k	
108	l	
109	m	
110	n	
111	o	
112	p	
113	q	
114	r	
115	s	
116	t	
117	u	
118	v	
119	w	
120	x	
121	y	
122	z	
123	{	(left brace)
124	¦	
125	}	(right brace)
126	~	(tilde)
127	DEL	(appears on screen as a blank)
128-159	(user-defined)	

EXERCISE 12-2

Using the Statements HCHAR and VCHAR (Drawing a 10 × 10 Square)

YOUR ACTION

1. Type NEW and enter the program as shown.

2. RUN the program several times.

3. Type program lines shown (do not type NEW).

4. Before you RUN the program with these lines added, write what you expect to see.

   _____

   _____

   _____

   _____

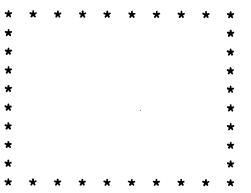
5. RUN the program.

6. Explain what happened and why.

7. LIST your program and make certain you understood it.

DISPLAY

```
>10 CALL CLEAR
>20 CALL HCHAR (14,5,42,10)
>30 CALL VCHAR (15,5,42,9)
>40 CALL HCHAR (23,6,42,9)
>50 CALL VCHAR (14,14,42,9)
```



```
>25 FOR DELAY = 1 TO 500
>27 NEXT DELAY
>35 FOR DELAY = 1 TO 500
>37 NEXT DELAY
>45 FOR DELAY = 1 TO 500
>47 NEXT DELAY
```

Note:

○ Line 20 displays an "*" (code 42) starting at Row 14, Col 5 and repeats it 10 times horizontally across Row 14.

Line 3Ø displays an "*" starting at Row 15, Col 5 and repeats it 9 times vertically down Col 5.

Line 4Ø displays an "*" starting at Row 23, Col 6 and repeats it 9 times horizontally across Row 23.

Line 5Ø displays an "*" starting at Row 14, Col 14 and repeats it 9 times vertically down Col 14.

EXERCISE 12-3

"Flashy" Characters

YOUR ACTION	DISPLAY
1. Type NEW and enter the program shown.	>1Ø CALL CLEAR >2Ø CALL HCHAR(13,1Ø,1Ø4) >3Ø CALL HCHAR(13,11,1Ø1) >4Ø CALL HCHAR(13,12,1Ø8) >5Ø CALL HCHAR(13,13,1Ø8) >6Ø CALL HCHAR(13,14,111) >7Ø FOR D = 1 TO 3ØØ >8Ø NEXT D >9Ø CALL CLEAR >1ØØ FOR D = 1 TO 2ØØ >11Ø NEXT D >12Ø GOTO 2Ø
2. RUN the program (press FCTN and CLEAR to stop).	HELLO ↑ (flashing)

Note:

○ Line 2Ø displays an "H" (code 1Ø4) at Row 13, Col 1Ø.

Line 3Ø displays an "E" (code 1Ø1) at Row 13, Col 11.

Line 4Ø displays an "L" (code 1Ø8) at Row 13, Col 12.

Line 5Ø displays an "L" (code 1Ø8) at Row 13, Col 13.

Line 6Ø displays an "O" (code 111) at Row 13, Col 14.

Lines 7Ø and 8Ø cause the program to pause after printing.

Line 9Ø clears the screen.

Lines 1ØØ and 11Ø delay the program after clearing.

Line 12Ø repeats the process.

○ The program above shows one way to create a flashing graphic. The steps are very simple:

1. Print the character (Lines 2Ø-6Ø).
2. Delay the program (Lines 7Ø and 8Ø).
3. Clear the screen (Line 9Ø).
4. Delay the program again (Lines 1ØØ and 11Ø).
5. Repeat the process (Line 12Ø).

12.9
CALL CHAR Statement

- Allows you to define or create your own special graphic characters. (A standard set of character codes 32-127 has already been defined in Section 12.8.)
  - — Also allows you to establish additional characters using codes 128-159. These codes are identified as "user-defined" characters.
- Format: CALL CHAR (Char Code, Pattern-Identifier)
  - — Where "Char Code" specifies the code of the character you wish to define and must be a value between 32 and 159, inclusive.
  - — The "Pattern-Identifier" is a 16-character string expression that specifies the pattern of the character you want to use in your program. This string expression is a coded representation of the 64 dots that make up a character position on the screen. These 64 dots comprise an 8 × 8 grid or dot matrix as shown on the next page greatly enlarged.
  - — To create a character, you must tell the computer which of the 64 dots of the 8 × 8 matrix to turn on and which to leave off. You do this only when you are creating nonstandard or new characters. Standard characters (A-Z and Ø-9) are automatically programmed so that the computer turns on the appropriate dots to produce the desired images.
- Example

```
CALL CHAR (128, "FFFFFFFFFFFFFFFF")
```

space

Character code you are defining

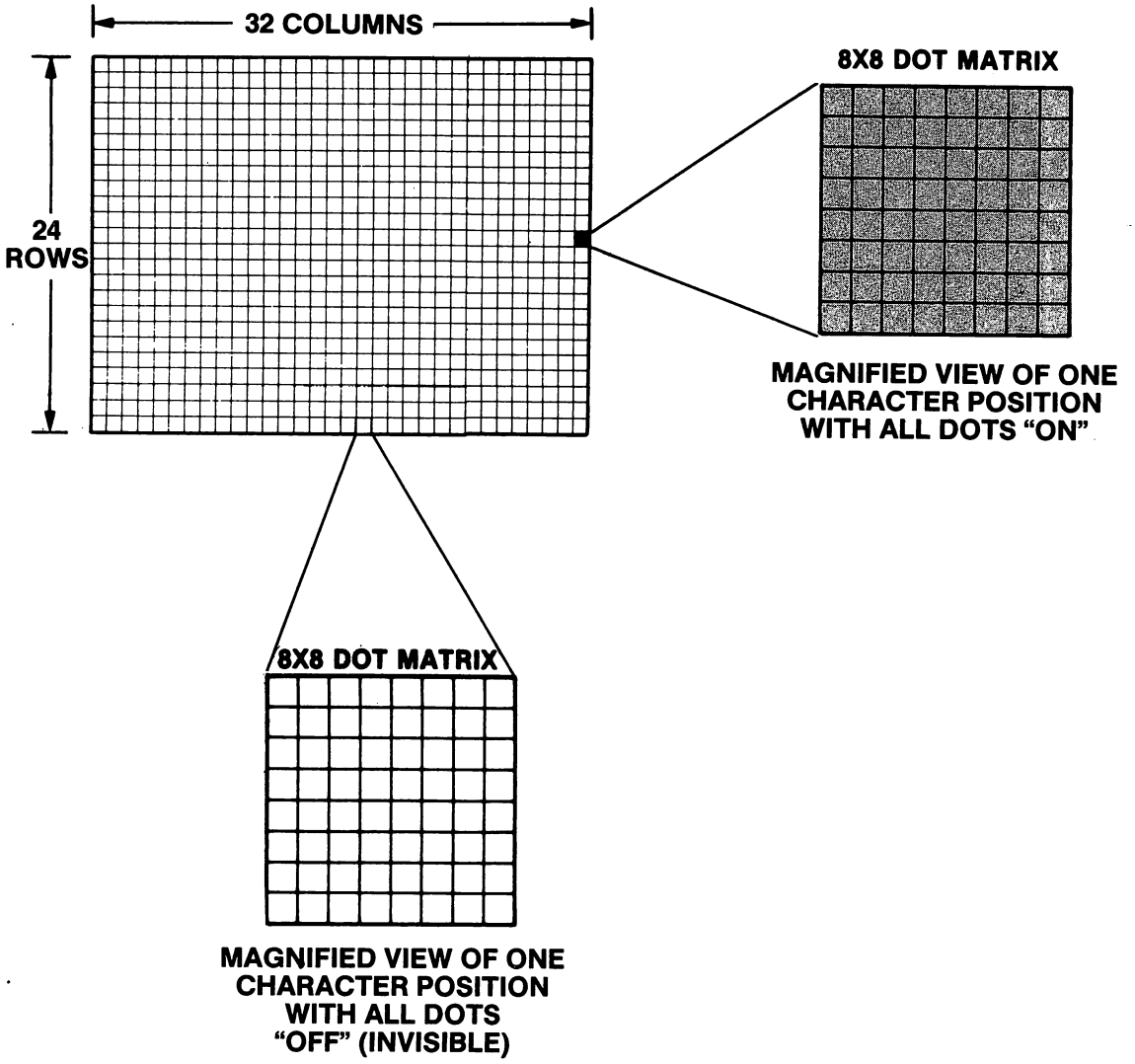
16-character string expression

Note:

○ Character code 128 is defined in this statement as a lighted dot by using the string expression "FFFFFFFFFFFFFFFF," which happens to be a shorthand code for turning all dots on.

12.10

Illustration of Dot Matrix for Two Character Positions



Note:

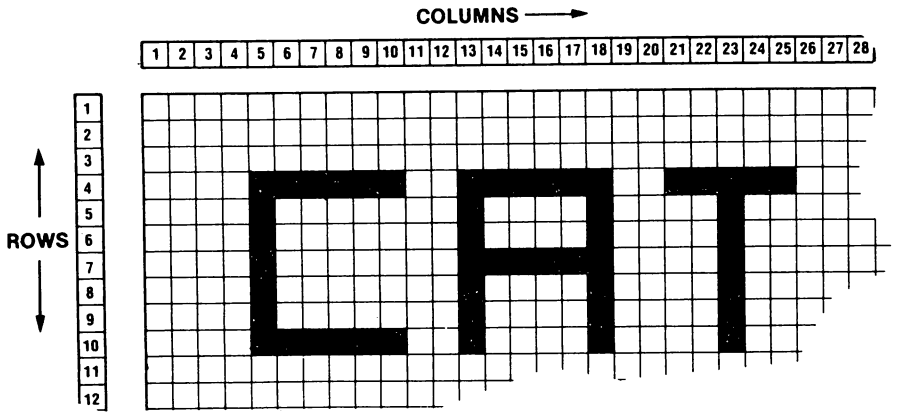
○ The character code for turning all dots "on" is "FFFFFFFFFFFFFFFF"

or

CALL CHAR(128,"FFFFFFFFFFFFFFFF")

EXERCISE 12-4

Drawing Large-Size Letters with TI-99/4A



YOUR ACTION	DISPLAY
1. Type NEW and enter the program shown.	>10 CALL CLEAR >20 CALL CHAR(128,"FFFFFFFFFF FFFFFF") >30 CALL HCHAR(4,5,128,6) >40 CALL VCHAR(5,5,128,6) >50 CALL HCHAR(10,6,128,5) >60 CALL HCHAR(4,13,128,6) >70 CALL VCHAR(5,13,128,6) >80 CALL VCHAR(5,18,128,6) >90 CALL HCHAR(7,14,128,4) >100 CALL HCHAR(4,21,128,5) >110 CALL VCHAR(5,23,128,6)
2. RUN the program.	(CAT should appear on screen)

Note:

○ You should draw the picture on the worksheet on page 163 first and use it as a guide for drawing graphic characters with the TI-99/4A.

○ Line 20 defines CHAR CODE 128 as a "■".

Lines 30-50 draw the "C."

Lines 60-90 draw the "A."

Lines 100-110 draw the "T."

12.11
TAB Function

- Specifies the starting position on the PRINT line for the next PRINT item.
- Operates much like a typewriter Tab key.
- Format: TAB (N) where N is a number in the range from 1 to 28.
- Example:

```
10 PRINT TAB (5); "TABBED 5"
```

Note:

- The PRINT line on the screen has 28 columns as compared to 32 columns for the "graphics line."
- The TAB function always counts from Column 1 (the leftmost PRINT position on the line) regardless of where and how many times it appears in the PRINT statement.
- For best results, you should use a semicolon before and after a TAB function if you have more than one TAB on a PRINT line.

12.12
TAB Example

YOUR ACTION	DISPLAY
1. Type and enter the program shown.	>10 CALL CLEAR >20 PRINT TAB (5);"STUDENT'S"; TAB (18);"STUDENT'S" >30 PRINT TAB (5);"NAME";TAB(1 8);"GRADE" >35 PRINT >40 PRINT TAB (5);"ADRIENNE";T AB (18);98 >45 PRINT >50 PRINT TAB (5);"ALYCE";TAB(18);100
2. RUN the program.	STUDENT'S NAME STUDENT'S GRADE ADRIENNE 98 ALYCE 100 ** DONE **

Note:

○ TAB function can be used to set up your results in a column format.
— All students' names would start printing at Column 5, TAB (5), and the students' grades would start printing at Column 18. Notice that the numbers 98 and 100 are actually printed in Column 19 because there is a space in front of a positive number.

SUMMARY

Graphics and TAB

- CALL CHAR statement allows you to define your own special graphics characters.
- CALL HCHAR displays a character anywhere on the screen and, optionally, repeats it horizontally.
- CALL VCHAR also displays a character anywhere on the screen but, optionally, repeats it vertically.
- TAB function specifies the starting position on the PRINT line for the next PRINT item.

ASSIGNMENT 12-1

Those of you who would really like to learn more about graphics should read the following pages in the *TI-99/4A User's Reference Guide:* pages II. 76 - 83 and III. 26 - 34. Also read Chapter 5, "Computer Graphics" (pages 96-127), in TI's *Beginner's BASIC* book.

PRACTICE 21

Graphics

Using solid blocks (CHAR CODE="FFFFFFFFFFFFFFFF"), write a program that will do the following:

1. Draw an 8x8 square starting five blocks from the left of the screen and five blocks from the top.
2. At the position ten blocks from the left side of the screen and ten blocks from the top, draw a letter C that is 7 blocks long and 5 blocks wide.
3. Try other pictures or letters (if you have the time).

PART 13

Arrays

What You Will Learn

1. To explain the purpose of using arrays.
2. To set up one- and two-dimensional numeric arrays.
3. To explain the purpose and use of the terms **DIM, A(3), A(2,3), DIM A(10), DIM DB(7,5).**
4. To develop, enter, and RUN programs using numeric arrays.

13.1
An Array

What is an array?

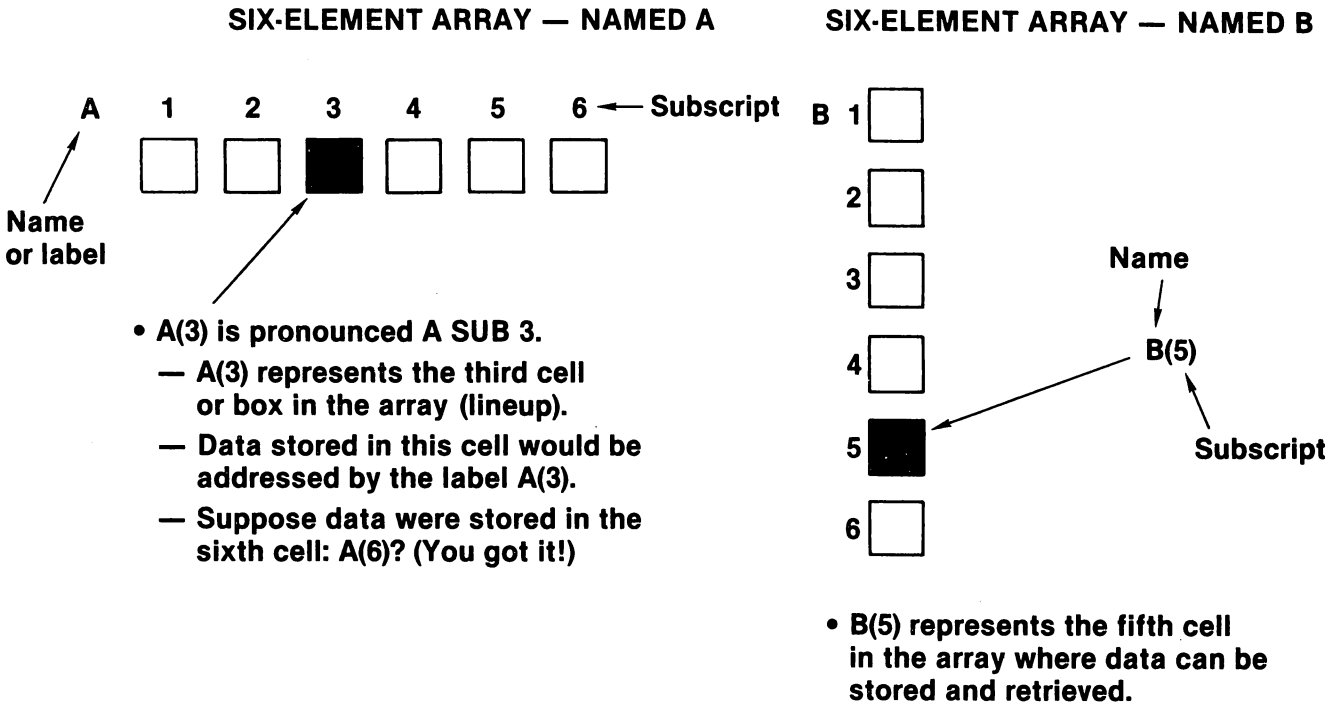
- An array is a lineup, an arrangement, or an orderly grouping of things.

Why use an array?

- We use it when we wish to have more variables available in a program.
  - — Although TI BASIC permits the use of many variables for numerics, sometimes thousands of variables are required for storing and retrieving pieces of data.
  - — The array allows you to arrange your data so that it can be stored and retrieved easily.

13.2

One-Dimensional Array Illustration



Note:

○ A and B are optional names. Any valid variable name can be used to name an array in TI BASIC.

13.3

One-Dimensional Array Program

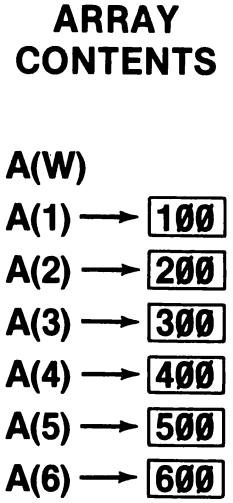
YOUR ACTION	DISPLAY
1. Type in these program lines.	>10 DATA 100,200,300,400,500,600 >20 FOR W=1 TO 6 >30 READ A(W) >40 NEXT W >50 FOR W=1 TO 6 >60 PRINT W,A(W) >70 NEXT W
2. RUN the program.	1 100 2 200 3 300 4 400 5 500 6 600

Note:

- ◯ Lines 20-40 store the data in array A(W).
- ◯ Lines 50-70 retrieve the data from array A(W) and PRINT the data.

13.4

One-Dimensional Array Program Analysis



- Above is an illustration of what happens after data are stored in array A(W). Note that in location A(1), the first data element (100) is stored. In location A(2), the second data element (200) is stored, and so on until the sixth data element (600) is stored in location A(6). Remember that Line 10 of the program contained the data elements that were read using Lines 20 through 40.

13.5

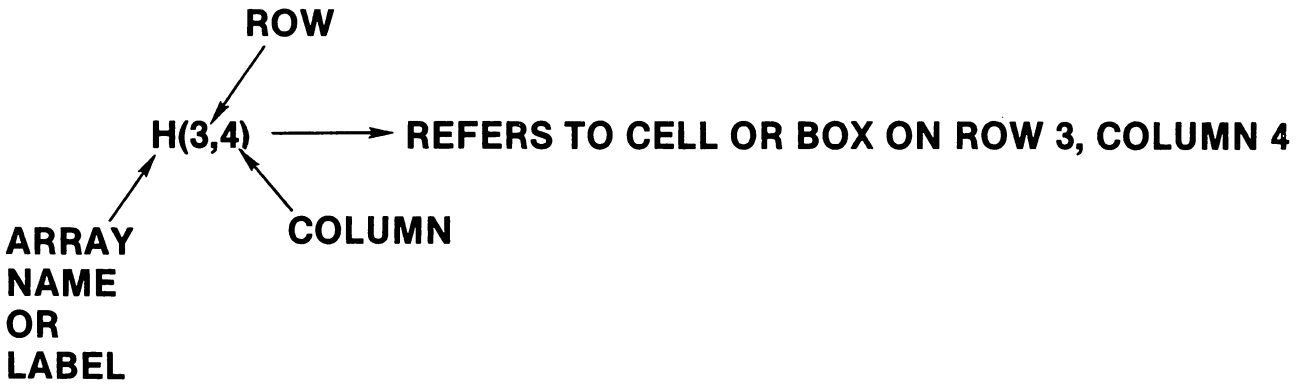
Two-Dimensional Array Illustration

COLUMN

ROW

H	1	2	3	4	5	6
1	11	12	13	14	15	16
2	21	22	23	24	25	26
3	31	32	33	34	35	36
4	41	42	43	44	45	46
5	51	52	53	54	55	56
6	61	62	63	64	65	66

36 ELEMENT ARRAY (MATRIX)
(NAMED H)



EXERCISE 13-1

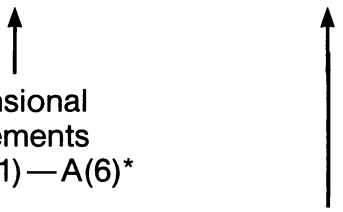
Fill in the blanks using the matrix on page 184.

LABEL	ROW	COLUMN	CONTENTS
H(1,1)	_____	_____	_____
H(4,5)	_____	_____	_____
H(3,3)	_____	_____	_____
H(2,3)	_____	_____	_____
H(6,6)	_____	_____	_____
H(1,6)	_____	_____	_____
H(2,4)	_____	_____	_____
H(4,4)	_____	_____	_____

13.6
DIM Statement

- DIM
  - — Lets you set the depth (number of elements allowed per dimension).
  - — If no DIM statement is used, a depth of 11 (subscripts Ø-1Ø) is allowed for each dimension of each array used.
  - — DIM statements may be placed anywhere in your program.
- Example:

```
1Ø  DIM  A(6),        B(2,3),        C(21)
```



Sets a one-dimensional array A with 6 elements A(Ø)—A(5) or A(1)—A(6)*

Sets a two-dimensional array B with 3 rows (numbered Ø-2) and 4 columns (numbered Ø-3)

Sets a one-dimensional array with 21 elements A(Ø)—A(2Ø) or A(1)—A(21)*

*If A(Ø) is not used.

13.7

Checkbook Array Example

- Consider the following table of checkbook information:

Check #	Date Written	Amount
100	6/5/83	$ 15.50
101	6/7/83	25.00
102	6/15/83	145.00
103	6/22/83	65.00
104	6/30/83	211.00
105	6/30/83	79.50

- Note that every item in the table may be specified by reference to two numbers: the row number and the column number. For example, Row 3, Column 3 refers to the amount $145.00.
- The above table can be set up in a 6x3 array or matrix.

CK	1	2	3
1	100	60583	15.50
2	101	60783	25.00
3	102	61583	145.00
4	103	62283	65.00
5	104	63083	211.00
6	105	63083	79.50

Note:

- ○ The date is recorded in the form mmddyy, where mm = month number, dd = day, and yy = the last two digits of year.
- ○ Since CK is a numeric array, alphanumeric characters such as dashes cannot be stored.

13.8

Checkbook Array Program — Setting up the Array

YOUR ACTION	DISPLAY
1. Type in these lines:	>5 CALL CLEAR >10 DIM CK(6,3) >20 FOR ROW=1 TO 6 >30 FOR COL=1 TO 3 >40 READ CK(ROW,COL) >50 NEXT COL >55 NEXT ROW >60 DATA 100,60583,15.50 >70 DATA 101,60783,25.00 >80 DATA 102,61583,145.00 >90 DATA 103,62283,65.00 >100 DATA 104,63083,211.00 >110 DATA 105,63083,79.50 >120 FOR ROW=1 TO 6 >130 SUM=SUM+CK(ROW,3) >140 NEXT ROW >150 PRINT "TOTAL OF CHECKS $"; >160 PRINT SUM
2. Type RUN.	TOTAL OF CHECKS $ 541

Note:

- ○ Line 10 sets up the dimension of the array. DIM CK(6,3) sets up an array with 6 rows and 3 columns.
- ○ Lines 20-55 read the values into array CK.
- ○ Lines 60-110 contain the values of the array in DATA statements.
- ○ Lines 120-140 add up all the checks written.

13.9

Checkbook Array Program — Manipulating the Array

YOUR ACTION | **DISPLAY**

3. **Do not type in NEW.**

4. Add these steps to your program:

```
>200 INPUT "LIST CHECKS WRITT
  EN ON (MM DD YY)":DT
>205 PRINT
>210 PRINT "CHECKS WRITTEN ON
  ";DT;"ARE LISTED BELOW:"
>215 PRINT
>220 PRINT "CHECK #","AMOUNT"
>230 FOR ROW=1 TO 6
>240 IF CK(ROW,2)<>DT THEN 25
  0
>245 PRINT CK(ROW,1),CK(ROW,3
  )
>250 NEXT ROW
```

5. Type RUN.
   (Enter 63083 for INPUT.)

```
TOTAL OF CHECKS $ 541
LIST CHECKS WRITTEN ON (MM D
D YY)■

CHECKS WRITTEN ON 63083
ARE LISTED BELOW:

CHECK #     AMOUNT
  104         211
  105         79.5
```

Note:

○ In Line 240 "<>" means "not equal to."

ASSIGNMENT 13-1

Read pages II.109 through II.112 in the *TI User's Reference Guide.*

SUMMARY

Arrays

- A2 is not the same as A(2).
  — A2 is an ordinary variable.
  — A(2) is a subscripted variable.
- Any time you have a subscript larger than 1Ø (depth of 11), you must use a DIM statement.
  — Example:
  1Ø DIMA (25), B(17, 18)
- One-Dimensional Array

  — A(3) is pronounced A SUB 3.
  (Subscript; Name)
- Two-Dimensional Array (Matrix)

  — H(3,4) refers to cell or box on Row 3, Column 4.
  (Row; Name; Column)

PRACTICE 22
Arrays

1. Write a program to read the following numbers into an array and then PRINT them out:
   676 15Ø 175 188 19Ø 277 876 976 912 544
2. Change the program to find the sum and average of the 1Ø numbers given.
3. Label the answer: THE SUM IS _____, and THE AVERAGE IS _____.

PRACTICE 23
One-Dimensional Array

1. Suppose we had the following results of a quiz given to a class of 1Ø students:

Student #	1	2	3	4	5	6	7	8	9	1Ø
Student's Grade	75	85	95	87	1ØØ	77	83	69	98	88

   a. Using a one-dimensional array, write a program to find the class average.
   b. Add the necessary program lines to find the highest grade and the lowest grade.
   c. Have the program PRINT: CLASS AVERAGE IS _____, HIGHEST GRADE IS _____, and LOWEST GRADE IS _____.
   d. Enter the program and RUN it several times.

PART

INT(X), ABS(X), and RND Functions

14

What You Will Learn

1. To explain the purpose and use of **INT(X), ABS(X),** and **RND** functions.
2. To explain the purpose and use of the key word RANDOMIZE.
3. To write, RUN, and analyze programs using the INT(X), ABS(X), and RND functions.

14.1
INT(X) Function

- INT(X), or integer function, allows you to round off any number, large or small, positive or negative, into a whole number (or integer).
- INT(X) means
  — If X is a positive number, then the largest whole number can be found by chopping off the decimal part.

    Example:
    INT (5.7) = 5
    INT (Ø.7) = Ø

  — If X is a negative number, the largest whole number can be found by moving down to the next lowest whole number (that is, by making a negative number more negative).

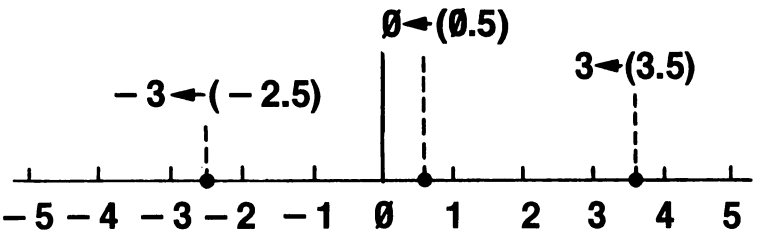
    Example:
    INT (−Ø.6) = −1 INT (−3.14) = −4
    INT (−Ø.2) = −1 INT (−7.28) = −8

Note:

○ The PRINT statement can be used for the above. For example, PRINT INT (5.7), then press the ENTER key.

EXERCISE 14-1
INT(X)

Graphic Representation



For negative numbers:
Move to next lowest whole number

For positive numbers:
Chop off decimal part

X	INT(X)
Ø.5	_____
–1.7	_____
2.345	_____
–Ø.8	_____
Ø	_____
3.1415	_____
76.14	_____
–1Ø.35	_____

14.2

INT(X) Function — Rounding $$

YOUR ACTION	DISPLAY
1. Type and enter this program.	>10 A =20/3 >20 PRINT"$";A
2. Now RUN.	$ 6.666666667
3. Add Line 15 to program as shown.	15 A =INT(100*A+.5)/100
4. Now RUN the program.	$ 6.67

Note:

○ In Line 15 we multiply by 100, add .5, take the INT (which is now 667), and then divide 667 by 100; 667/100 is 6.67, which is what we want — two decimal places.

ASSIGNMENT 14-1

INT(X)

1. Type NEW and enter this program for finding the area of a circle:

```
5 CALL CLEAR
10 REM AREA OF A CIRCLE
15 REM AREA=PI*R^2
20 INPUT "WHAT IS THE RADIUS?":R
30 PI=3.14159
40 A=PI*R^2
50 PRINT "THE AREA IS";A
```

2. RUN the program several times to make sure it works.
3. Change the program to suppress (chop off) all of the numbers to the right of the decimal point. (RUN the program to make sure it works.)
4. Change the program to make the answer accurate to one decimal place. (For example, if R = 1, then Area (A) = 3.1.)

14.3
ABS(X) Function

- ABS(X) = Abbreviation for absolute value of X
- Examples:

ABS (12) = 12	ABS (− 1Ø) = 1Ø
ABS (Ø) = Ø	ABS (−357) = 357

Note:

○ ABS (25 − 1Ø) = ABS (1Ø − 25) = 15

ASSIGNMENT 14-2
ABS(X)

YOUR ACTION	DISPLAY
1. Type and enter the program shown.	>1Ø INPUT "TYPE ANY POSITIVE OR NEGATIVE #": N >2Ø X = ABS (N) >3Ø PRINT "N","X" >4Ø PRINT N,X
2. RUN the program several times using both positive and negative numbers.	

Note:

○ Regardless of the number you INPUT as N, the absolute value of X is the same number without the sign.

14.4

RND Function and RANDOMIZE Statement

- RND or random number function causes the computer to give you a "surprise" number.
  — It's as though the computer spins a wheel of chance.
  — It's like pulling a number out of a hat.
  — It's unpredictable!
- RANDOMIZE is a complete program statement that "reseeds" the random number generator.
  — Each time you use RND the computer uses an internal seed number to produce the desired random number. Therefore, it is a good idea to set the seed number (reseed) to an unpredictable value in any program that uses random numbers.

YOUR ACTION	DISPLAY
1. Type in these lines.	>5 RANDOMIZE >1Ø PRINT RND >2Ø GOTO 1Ø
2. Type RUN. (Press FCTN and CLEAR to stop.)	(Screen should show many random decimal numbers.)

Note:

○ RND alone will produce numbers between Ø and 1.

14.5

RND Function (Generating Integers)

- RND alone will produce a number between Ø and 1.
- To generate integers between 1 and X, use the form

  LET N=INT(X*RND)+1

Note:

○ The general form for generating integer random numbers may seem complicated at first; just remember that X is the highest random number you wish generated.

YOUR ACTION	DISPLAY
1. Type NEW and enter these lines.	>5 RANDOMIZE >10 X=20 >20 FOR N=1 TO 10 >30 PRINT INT (X*RND)+1; >40 NEXT N
2. Type RUN.	(Screen will show ten random numbers between 1 and 20.)
3. Type RUN.	(Screen will show another ten random numbers between 1 and 20.)
4. Change these lines.	>10 INPUT "VALUE FOR X":X >20 FOR N=1 TO 50
5. RUN Several times. (Enter some number between 5 and 100 at the INPUT.)	(Screen will show fifty random numbers between 1 and the number you INPUT.)
6. LIST the program.	5 RANDOMIZE 10 INPUT "VALUE FOR X?":X 20 FOR N=1 TO 50 30 PRINT INT(X*RND)+1 40 NEXT N

ASSIGNMENT 14-3

Coin Toss Program

1. Type in the following program:

```
5 CALL CLEAR
6 REM COIN TOSS PROGRAM
7 RANDOMIZE
10 REM T=TAILS, H=HEADS
15 T=0
20 H=0
30 INPUT "HOW MANY TIMES SHALL I FLIP THE COIN?":N
40 CALL CLEAR
50 PRINT "I'M FLIPPING THE COIN...STAND BY"
60 FOR K=1 TO N
70 X=INT(2*RND)+1
80 IF X=1 THEN 100
90 H=H+1
95 GOTO 110
100 T=T+1
110 NEXT K
120 CALL CLEAR
130 PRINT "HEADS","TAILS"
140 PRINT
145 PRINT H,T
150 PRINT 100*H/N;"%",100*T/N;"%"
155 PRINT
160 PRINT "TOTAL FLIPS =";N
```

2. RUN this program several times and discuss the results.

Note:

○ Lines 15 and 20 initialize the counters (H and T) to zero.
Line 60 begins the FOR-NEXT statement and executes "N" times.
Line 70 generates a random number (either a 1 or a 2).
In Line 90, "heads" are counted.
In Line 100, "tails" are counted.
Line 130 prints the headings.
Line 140 prints the values of H and T.
Line 150 calculates and prints the percentage of heads, percentage of tails.
Line 155 provides a space for better appearance.

ASSIGNMENT 14-4

Guess the Number Program

1. Type in the following program:

```
5 REM GUESS THE NUMBER GAME
10 RANDOMIZE
20 CALL CLEAR
30 X=INT(10*RND)+1
40 INPUT "GUESS A NUMBER BETWEEN 1 & 10:":N
50 IF X=N THEN 110
60 IF X<N THEN 90
70 PRINT "HIGHER"
80 GOTO 40
90 PRINT "LOWER"
100 GOTO 40
110 PRINT "THAT'S RIGHT!"
120 FOR J=1 TO 2000
130 NEXT J
140 GOTO 10
```

2. RUN the program. (To stop the program, use the FCTN and CLEAR keys.)

3. Analyze the program.

   Line 10 _____ the random number generator.

   Line 30 is the _____ generator.

   Line 40 allows the user to _____ a number.

   Lines 50 and 60 are _____ statements that compare
   (conditional, unconditional)

   the random number _____ with the INPUT number _____.
   X,N X,N

   Lines 100, 70 and 90 are PRINT statements that guide the player.
   Why does Line 140 GOTO Line 10, and why do Lines 80 and 100 GOTO Line 40? Why were only two IF-THEN statements necessary to compare the INPUT with the random number?

4. Modify (change) the program to pick a number between 1 and 1ØØ, and RUN this program several times.

SUMMARY

INT(X), ABS(X), RND, RANDOMIZE

- INT(X) — Provides the integer or whole-number value of X.
  - — If X is a positive number, it chops off the decimal part.
  - — If X is a negative number, it rounds it down to the next lowest whole number (e.g., INT (−Ø.6) = −1).
- ABS(X) — Provides the absolute value of X (i.e., X is that same number without the sign).
- RND — Causes the computer to give you a random number.
  - — INT (X * RND)+1 gives you a random number from 1 to X inclusive.
- RANDOMIZE — A complete program statement that reseeds the random number generator.
  - — It is used at the beginning of any program that uses random numbers to ensure that the same sequence of random numbers are not generated each time the program is RUN.

PRACTICE 24
INT(X)

1. Fill in the blanks with the appropriate INT(X):

X	INT(X)
Ø.7	_____
−2.5	_____
6.365	_____
−Ø.8	_____
−1Ø.65	_____
Ø	_____
3.2425	_____
−7.61	_____
−Ø.3	_____
Ø.3	_____

2. The following program can be used for finding the area of a circle:

```
1Ø REM AREA OF A CIRCLE
15 REM A=3.14159*RΛ2
2Ø INPUT "THE RADIUS IS": R
25 INPUT "THE RADIUS IS IN (IN., FT, OR YD)": A$
3Ø A = 3.14159*RΛ2
4Ø PRINT "THE AREA IS"; A; "SQ. "; A$
```

   a. Enter and RUN the program several times to make certain it works.
   b. Change the program to suppress (chop off) all the numbers to the right of the decimal point (RUN the program to make sure it works).
   c. Change the program to make the answer accurate to one decimal place. (For example, if R = 1, then area (A) = 3.1).

PRACTICE 25
RANDOMIZE and RND

1. Write a program that will let you pick a random number between 1 and 1ØØ. The program should let you INPUT a number from the keyboard and provide the following clues on your guess.
   a. If the number you pick matches the number the computer picks, have the computer PRINT "RIGHT ON."
   b. If the number from the keyboard is too high, have the program print "LOWER."
   c. If the number from the keyboard is too low, have the program print "HIGHER."
   d. Enter the program and RUN it several times.

PART 15

Subroutines

What You Will Learn

1. To explain the purpose for using subroutines.
2. To explain the purpose and use of the key words **GOSUB, RETURN, ON-GOTO, ON-GOSUB.**
3. To develop, enter, and RUN programs using subroutines.

15.1
Subroutine

What Is It?

- A subroutine is a short program or routine that is built into a large program to do specific calculations or perform repetitive functions.

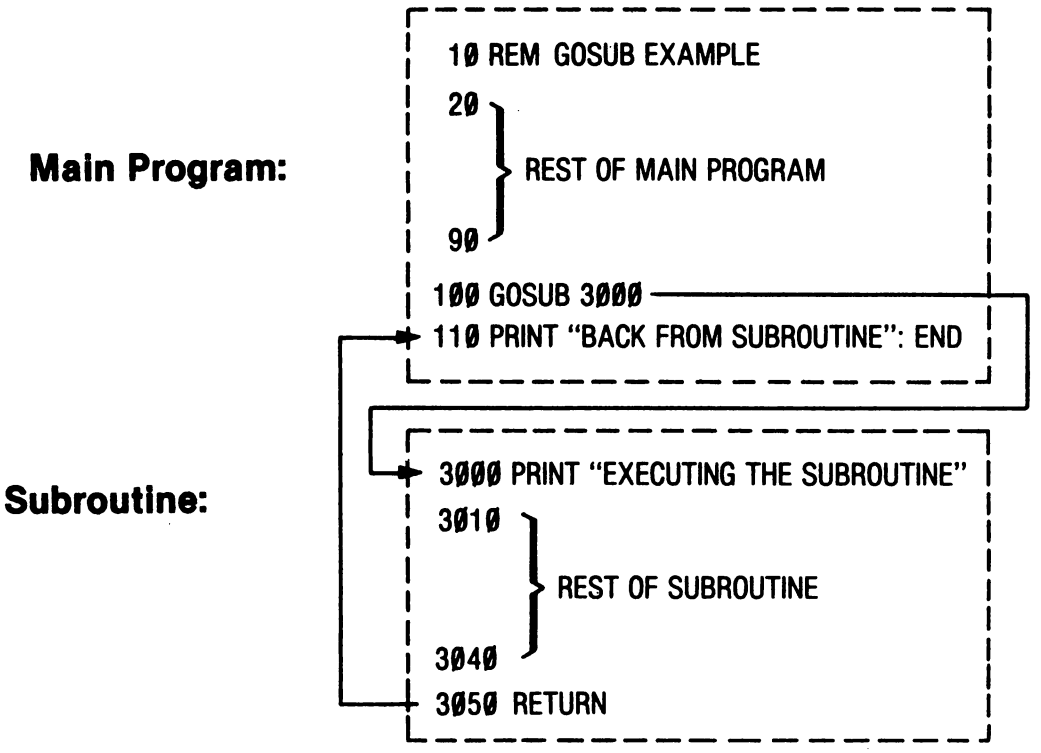
Why Use It?

- There are times when you need the same type of calculation at various points in your program, but instead of retyping the statements needed for this calculation each time, you can write a subroutine to perform the needed calculations.

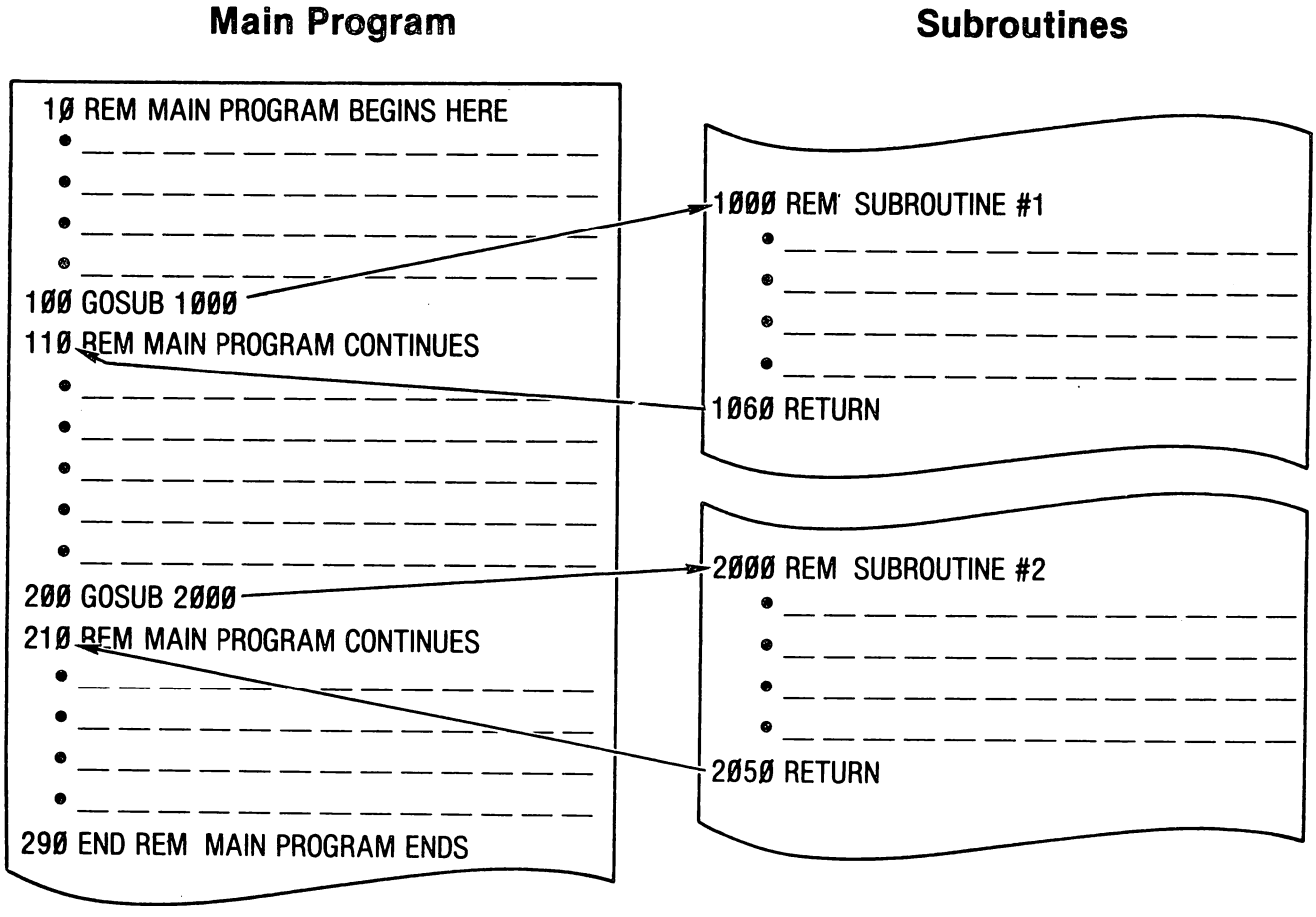
How Do You Call a Subroutine?

- To call or branch to a subroutine, use the GOSUB statement.
  - — The GOSUB XXXXX statement directs the computer to go to that line number and execute the program steps until it reaches the key word RETURN, which ends the subroutine.
  - — RETURN is always built into a subroutine and is used to tell the computer that the subroutine is finished. When it is finished, the control of the program is returned to the statement in the main program immediately following the most recently executed GOSUB.

15.2
Subroutine Example



15.3
Subroutine Illustration



15.4

Analysis of Subroutine Illustration

1. When the computer reaches the GOSUB in Line 1Ø, the program will branch to Line 1ØØ, which is the beginning of Subroutine #1.
2. After Subroutine #1 is executed and the RETURN (Line 1Ø6Ø) is reached, control is passed back to the main program (Line 11Ø). Note that Line 11Ø is the next higher number after the GOSUB that put it in the subroutine (Line 1Ø).
3. The computer continues through the main program to the GOSUB in Line 2Ø, which branches control to Subroutine #2 in Line 2ØØ.
4. After the subroutine is executed, the RETURN (Line 2Ø5Ø) passes the control back to Line 21Ø in the main program. (Note again that this is the next higher line number after the GOSUB in Line 2Ø.)
5. An END statement is included in the program (Line 29Ø) after the main program is finished to keep it from accidentally falling into the subroutine. We want the subroutines to be executed only when we call for them by a GOSUB.

15.5

Sample Program Using Subroutines (Temperature Conversion)

Main Program

```
10 REM TEMPERATURE CONVERSION PROGRAM
15 CALL CLEAR
20 INPUT "DO YOU WISH TO CONVERT C TO F (Y OR N)?":A$
25 PRINT
30 IF A$ = "Y" THEN 90
40 INPUT "DEGREES FAHRENHEIT ?":F
50 GOSUB 2000
55 PRINT
60 INPUT "HAVE YOU FINISHED (Y OR N)?":B$
70 IF B$ = "N" THEN 15
80 GOTO 120
90 INPUT "DEGREES CELSIUS ?":C
95 GOSUB 1000
100 GOTO 55
120 END
```

Subroutine #1

```
1000 REM C TO F CONVERSION ROUTINE
1010 F = (9/5)*C + 32
1015 PRINT
1020 PRINT C;"DEGREES C =";F;"DEGREES F"
1030 RETURN
```

Subroutine #2

```
2000 REM F TO C CONVERSION ROUTINE
2010 C = (F-32) * (5/9)
2015 PRINT
2020 PRINT F;"DEGREES F =";C;"DEGREES C"
2030 RETURN
```

15.6

Analysis of Sample Program Using Subroutines

1. Lines 1Ø through 12Ø comprise the main program.
2. Line 2Ø is an INPUT statement asking the user if he wants to convert from C to F. A yes answer ("Y") means the user wants to convert from Celsius to Fahrenheit.
3. Line 3Ø tests the answer to the INPUT and branches either to Line 9Ø or to Line 4Ø.
4. Line 4Ø allows the user to INPUT the degrees Fahrenheit.
5. Line 5Ø branches to the subroutine on converting F to C (Line 2ØØØ).
6. Line 6Ø asks the user if he is finished. If the user answers "N" then the program will clear the screen (Line 15) and start the main program over again. If the user answers "Y" then the program will end (Line 12Ø).
7. Line 9Ø is similar to Line 4Ø. It allows the user to INPUT the degrees Celsius.
8. Line 95 branches to the subroutine on converting C to F (Line 1ØØØ).
9. Line 1ØØ routes the program to Line 55 where a line feed is executed for better output appearance and the user is asked if he is finished.
10. Line 1ØØØ through Line 1Ø3Ø contain the subroutine for converting Celsius to Fahrenheit and PRINTing the answer. Line 1Ø3Ø RETURNs control to Line 1ØØ in the main program.
11. Line 2ØØØ through Line 2Ø3Ø contain the subroutine for converting Fahrenheit to Celsius and PRINTing the answer. Line 2Ø3Ø RETURNs control to Line 55 in the main program.

15.7
Subroutine Exercise

```
10 PRINT "THIS IS";" ";
20 GOSUB 1000
30 PRINT "OF HOW";" ";
40 GOSUB 2000
50 PRINT "WORKS"
60 END
1000 PRINT "AN EXAMPLE";" ";
1010 RETURN
2000 PRINT "A SUBROUTINE";" ";
2010 RETURN
```

1. Analyze the program and write the message. _____
2. Now type and enter the program.
3. RUN the program. Does it agree with your message?

ASSIGNMENT 15-1

Analyze the program below and write the message:

```
10 LET B = 10
20 GOSUB 2000
30 B = B + 5
40 GOSUB 2000
50 B = B + 10
60 GOSUB 2000
99 END
2000 REM SUBROUTINE
2010 IF B < 12 THEN 2045
2020 IF B = 25 THEN 2070
2030 PRINT "PRIME"
2040 GOTO 2080
2045 CALL CLEAR
2050 PRINT "LEEDS"
2060 GOTO 2080
2070 PRINT "COMPUTERS"
2080 RETURN
```

Message _____

15.8

Conditional Branching (The Long Way!)

YOUR ACTION	DISPLAY
1. Type NEW and enter this program.	>5 CALL CLEAR >10 INPUT "TYPE A NUMBER FROM 1 TO 3":N >20 IF N = 1 THEN 110 >30 IF N = 2 THEN 130 >40 IF N = 3 THEN 150 >50 PRINT "HEY, I WANT A NUMBER FROM 1 TO 3!" >60 GOTO 10 >110 PRINT "N = 1" >120 END >130 PRINT "N = 2" >140 END >150 PRINT "N = 3" >160 END
2. Before you RUN the program, analyze it. Can you predict what will happen when you RUN it? (I sure hope you can by now!)	
3. RUN the program several times until you feel comfortable with it.	

15.9

Conditional Branching (The Short Way) ON-GOTO

YOUR ACTION | **DISPLAY**

4. Erase Lines 20, 30, and 40. Simply type in each line number and press ENTER.

5. Type this line:

```
>20 ON N GOTO 110,130,150
```

6. LIST your program.
(If your program does not look like this, fix it!)

```
5 CALL CLEAR
10 INPUT "TYPE A NUMBER FROM
 1 TO 3 ":N
20 ON N GOTO 110,130,150
50 PRINT "HEY, I WANT A NUMB
ER FROM 1 TO 3!"
60 GOTO 10
110 PRINT "N = 1"
120 END
130 PRINT "N = 2"
140 END
150 PRINT "N = 3"
160 END
```

7. RUN the program with the following INPUTs:
1.5
1.8
1.2
.8
2.6
3.99
.4

```
N = 2
N = 2
N = 1
N = 1
N = 3
* BAD VALUE IN 20
* BAD VALUE IN 20
```

Note:

○ The ON-GOTO statement *rounds* the value of the INPUT. If the rounded value of the INPUT is less than 1 or greater than the number of line numbers listed (in this case, 3), the program will stop running and you will get the error message shown. You should anticipate this and use IF-THEN statements to check the INPUT value whenever you plan on using ON-GOTO.

ASSIGNMENT 15-2

1. Type and enter the following program:

```
5 CALL CLEAR
10 INPUT "ENTER A NUMBER FROM 1 TO 5: ":N
20 IF N <> INT(N) THEN 10
30 IF N < 1 THEN 10
40 IF N > 5 THEN 10
50 ON N GOTO 100,110,120,130,140
100 PRINT "N = 1"
105 GOTO 10
110 PRINT "N = 2"
115 GOTO 10
120 PRINT "N = 3"
125 GOTO 10
130 PRINT "N = 4"
135 GOTO 10
140 PRINT "N = 5"
145 GOTO 10
```

Note:

- ○ Line 20 checks to see if the INPUT was an integer. If N does not equal the integer of itself, then the program swings back to the INPUT for another value.
- ○ Line 30 checks if N is less than 1. This is to make sure an error message does not occur.
- ○ Line 40 checks if N is greater than 5, also turning control back to Line 10 if it is, thus avoiding the error message.

2. RUN the program several times and record the following:

INPUT	OUTPUT

15.10
ON-GOSUB

- Has the form

  ON n GOSUB XXXX,YYYY,ZZZZ,...

  where n is a constant, a variable, or a variable expression.
- Works similarly to ON-GOTO except control branches to one of the *subroutines* specified by the line numbers in the line number list.
- If the value of n is less than 1 or greater than the number of line numbers in the list, an error message will occur.
- After one of the subroutines has been executed and a RETURN is encountered, control passes to the line following the ON-GOSUB statement.

ASSIGNMENT 15-3

1. Type in the following program:

```
5 RANDOMIZE
10 CALL CLEAR
20 PRINT "THIS PROGRAM GIVES A QUIZ."
30 PRINT "YOU HAVE A CHOICE:"
40 PRINT "1. ADD","2. SUBTRACT"
50 PRINT "3. MULTIPLY","4. DIVIDE"
60 INPUT "WHAT IS YOUR CHOICE? ":N
70 IF N <> INT(N) THEN 30
80 IF N < 1 THEN 30
90 IF N > 4 THEN 30
100 A=INT(10*RND)+1
110 B=INT(10*RND)+1
120 ON N GOSUB 500,600,700,800
130 INPUT "YOUR ANSWER? ":S
140 IF S=A1 THEN 180
150 PRINT "NO. THE ANSWER WAS";A1
160 INPUT "ANOTHER PROBLEM? ":A$
170 IF A$="YES" THEN 10
175 END
180 PRINT "VERY GOOD!"
190 GOTO 160
500 A1=A+B
510 PRINT A;"+";B;" = "
520 RETURN
600 A1=A-B
610 PRINT A;"-";B;" = "
620 RETURN
700 A1=A*B
710 PRINT A;"*";B;" = "
720 RETURN
800 A1=A/B
810 PRINT A;"/";B;" = "
820 RETURN
```

2. Change the program so that:
   - the problems generate larger numbers.
   - the answer to the subtraction problem is never negative.
   - the answer to the division problem is always an integer.
   - the person always gets 5 problems at a time.
   - the person receives a score of the number of correct answers.

SUMMARY

GOSUB, ON-GOSUB, ON-GOTO

- GOSUB XXXX
  - — Here the computer branches to the subroutine beginning at XXXX (the specified line number).
  - — The subroutine is executed until a RETURN statement is encountered.
  - — Program control returns to the statement that follows the GOSUB statement in the main program.
- ON n GOSUB XXXX,----,YYYY
  - — This is a conditional branching statement that sends control of the program to one of the subroutines specified in the line number list (XXXX,----,YYYY).
  - — The value of the test variable (n) determines which subroutine is executed. For example, if n=1, control is passed to the subroutine in line XXXX.
  - — Once a subroutine is being executed, the rules governing GOSUB are followed, i.e., a RETURN passes control to the statement following the ON n GOSUB.

- ON n GOTO XXXX,----,YYYY
  - — This statement works similar to ON n GOSUB except control does not pass to a subroutine but to another part of the main program.
  - — The value of n determines which line number is executed next.

Note:

○ n can be a constant, a variable, or a variable expression.

○ n must not be less than 1 or greater than the number of line numbers in the line number list or an error message will occur.

PRACTICE 26

Program to Convert Celsius to Fahrenheit and Vice Versa

1. Write a program that will do the following:
   a. Convert Celsius to Fahrenheit.
   b. Convert Fahrenheit to Celsius.
   c. Allow you to select either A or B above.
   d. Allow you to INPUT from keyboard.
   e. PRINT the answer as follows:

   ___*___ DEGREES CELSIUS = ___**___ DEGREES FAHRENHEIT

   or

   ___*___ DEGREES FAHRENHEIT = ___**___ DEGREES CELSIUS

   *Keyboard INPUT value
   **Calculated output value

PRACTICE 27

Program for Sample Profit/Loss Statement

1. When a product is sold for more than it costs, the seller receives a profit. When a product is sold for less than it costs, the seller takes a loss.

   Therefore: sell price − cost = profit or loss

   If we let: S = Sell price
   C = Cost
   U = No. of units
   P = Profit
   L = Loss

   Then: P (or L) = S*U − C*U

   a. Write a program that will compute the profit or loss for a business if the sell price and cost are known. (**Note:** Program should permit you to enter cost and sell price from the keyboard.)

   b. Have the computer PRINT the following:

   NO. OF UNITS _____
   UNIT PRICE ($) _____
   UNIT COST ($) _____
   TOTAL SALES ($) _____
   TOTAL COST ($) _____
   PROFIT/LOSS ($) _____
   % OF SALES _____

   c. RUN the program several times and record your answer.

EXTRA PRACTICE 1

Programming Mathematical Operators

1. Given two numbers A=25 and B=5:

   a. Write one program that will add, subtract, divide (A/B), multiply, and square the two numbers (A and B).

   b. The answer should PRINT as shown here:

   THE SUM OF A AND B IS _____ (your answer).
   THE DIFFERENCE BETWEEN A AND B IS _____ (your answer).
   THE QUOTIENT OF A AND B IS _____ (your answer).
   THE PRODUCT OF A AND B IS _____ (your answer).
   THE SQUARE OF A IS _____ (your answer).
   THE SQUARE OF B IS _____ (your answer).

EXTRA PRACTICE 2

Finding the Average

1. Write a program to find the average of three numbers.
2. Have the program PRINT: THE AVERAGE IS _____.
3. Add a program line to have the program PRINT the average of your # _____, your # _____, and your # _____ is (your answer). Example: THE AVERAGE OF 3, 4, AND 8 IS 5.

EXTRA PRACTICE 3

More Mathematical Operations

Write five separate programs to PRINT the answer to these problems (the answer should read 25 * 2 + 4 = 54, and so on.):

1. 25*2+4
2. 3∧2+4−2
3. 36/4*5
4. 28+4*6/8
5. (18−2)/3+4(6*3)+2∧3

EXTRA PRACTICE 4

Print Zones

Part I.

Write a program to PRINT the word "LEEDS" in the following ways:

	ZONE 1	ZONE 2
1.	LEEDS	LEEDS
2.	LEEDS	
3.		LEEDS

Part II.

Using page 76, type in the information as shown (LEEDSPRIME)...and so on.

1. Count the number of characters in the two zones. How many?
2. How many in Zone 1 _____; Zone 2 _____?

EXTRA PRACTICE 5

Area of Square and Volume of Cube

1. Write a program to solve the following problems. Label your answers.
   a. The side of a square is 27 inches. Find its area (area (A) = s^2).
   b. If the side of a cube is also 27 inches, find its volume (volume (V) = s^3).
2. Using INPUT statements, write a program to find the area of a square and volume of a cube.
   a. Solve the problems above (assume sides of a square and cube are equal).
   b. Using different lengths for the side, RUN the program again (assume that the sides of the square and the cube are equal).

EXTRA PRACTICE 6

Printing Tables of Numbers, Squares, and Cubes

1. Write a program to generate the first 25 numbers and PRINT their squares on the same line.

   Example:
   ```
   1     1
   2     4
   3     9
   4     16
   and so forth
   ```

2. Write a program to generate the first 25 numbers and PRINT their cubes on the same line.

   Example:
   ```
   1     1
   2     8
   3     27
   4     64
   and so forth
   ```

3. Write a program to generate all the numbers from 2Ø to 1 and PRINT the numbers, and their squares and cubes, on the same line and in four columns.

   Example:
   ```
   2Ø      4ØØ      8ØØØ      16ØØØØ
   19      361      6859      13Ø321
   18      324      5832      1Ø4976
   and so forth
   ```

EXTRA PRACTICE 7

Printing Three-Times and Nine-Times Tables

1. Write a program to generate the three-times table from 3 × 1 = 3 to 3 × 12 = 36. The printout should look exactly like this:

   ```
   3 * 1 = 3
   3 * 2 = 6
   3 * 3 = 9
   3 * 4 = 12
   and so forth
   ```

2. Write a program to generate the nine-times table from 9 × 1 = 9 to 9 × 12 = 1Ø8.

EXTRA PRACTICE 8

Two-Dimensional Array

1. Suppose we have a class of ten students. The course grade is based upon three quizzes, and the results for the class are as follows:

Student #	1	2	3	4	5	6	7	8	9	10
Quiz #										
1	88	41	100	88	79	76	86	90	85	100
2	75	52	65	57	98	86	96	91	86	92
3	71	47	75	77	86	96	85	92	97	82

a. Write a program to PRINT the following information:

Student #	*Course Avg./Student*	
1	?	
2	?	Computer to calculate and PRINT average
3	?	
4	?	

and so forth

Quiz #	*Course Avg./Quiz*	
1	?	
2	?	Computer to calculate and PRINT average
3	?	

HAYDEN BOOK COMPANY
a division of Hayden Publishing Company, Inc.
Hasbrouck Heights, New Jersey

ISBN 0-8104-6173-0